

# Implementing Enterprise-class Web services

*A White Paper for Architects and Developers*

## ***Mark Jones, Independent Consultant***

Mark Jones has spent the last 15 years moving up the integration food chain. He began his career with IBM's Information Network (later Advantis) mastering the world of EDI. His next adventure at education software company National Computer Systems thrust him squarely in front of the Internet juggernaut. Mark responded by launching one of the very first commercial Internet-based EDI initiatives in 1994. Mark eventually moved on to Mercator Software where he drove many of the technical innovations during the heyday of the EAI market. Mark believes Web services represent the next evolutionary step in the integration food chain and occasionally speaks and publishes his thoughts.

Email: [mjones20002@hotmail.com](mailto:mjones20002@hotmail.com).

## CONTENTS

---

The Need for Enterprise-class Web services.....	1
Web services vs. Enterprise-class Web services .....	1
Web services As Feature vs. Web services As Paradigm Shift .....	2
Adoption of Web services.....	3
Enterprise Web services Defined .....	5
Web services Platform (Provider and Requester) .....	6
Web services Broker .....	7
Web services Networks .....	8
Networked EAI Trading Partner Servers .....	9
Hub and Spoke "Value Added" Networks.....	10
Native Web services Networks .....	11
The Enterprise Web services Ecosystem .....	13
Examples of Web services Use Case Variables .....	14
RPC-Style Web services vs. Message-Style Web services.....	14
Intra vs. Inter-enterprise Web services .....	14
Use Cases .....	14
Order Status Portal .....	15
Supply Chain Automation .....	17
Summary .....	19

# The Need for Enterprise-class Web services

Web services are here. And, based on their widespread adoption, unprecedented cooperation amongst major vendors and rapid inclusion into the fabric of all things e-commerce, they appear to be here to stay. How well they live up to their lofty expectations and revolutionize the way businesses develop, deploy and share their information assets will be debated, documented and ultimately decided in the court of public opinion over the next few years. But, make no mistake about it Web services have definitely arrived.

The composition, role and value of Web services have been documented in great detail in numerous white papers. IBM, Microsoft and other notable industry giants have devoted significant resources to user community education on the value of Web services and promotion of their Web services offerings. Virtually every platform and application vendor has announced support for Web services and the core standards (SOAP, UDDI, WSDL and XML) are universally accepted. Perhaps most appealing, the Web services concept is fairly easy to understand with emphasis on simplicity and universal applicability. Web services have definitely arrived.

The real question, and the one that will be hotly debated over the next few years, is how big an impact Web services will have in any given enterprise and how quickly. Will Web services revolutionize the software industry as many are predicting, or provide yet another evolutionary step forward? What will the adoption rate be?

To answer these questions, it is critical to understand two key comparisons: 1) The difference between simply deploying a Web service and deploying an enterprise-class Web services architecture, and 2) Web services as a feature vs. Web services as a paradigm shift.

## Web services vs. Enterprise-class Web services

The first comparison addresses the “maturity” of the technical implementation. A simple Web services deployment shares characteristics typical of many new technology deployments within an enterprise such as department level prototypes, ad-hoc deployment procedures, inconsistent quality of service and general lack of coordination. Enterprise-class Web services, on the other hand, represent a level of technical and operational maturity consistent with the increasingly stringent requirements of today's enterprise. True enterprise-class deployments are characterized by:

- Physical scalability – refers to the ability to handle increasing volumes of usage without requiring drastic reconfiguration, testing and new hardware.
- Developmental scalability - addresses the ability to spread development projects among multiple teams and locations without compromising source integrity, reuse and test/QA/production promotion cycles.

- Management scalability - requires robust, precise monitoring tools that allow prediction, identification and resolution of an infinite number of deployed interfaces, objects or services. Mature technologies typically provide a combination of "lights out" monitoring tools (i.e. SNMP hooks) and real-time monitoring tools that provide a rich viewing and customization environment.
- Rollout scalability - is a relatively new dimension that deals with partner enablement. Given Web services' tremendous potential for business-to-business automation, it is imperative that an enterprise-class solution support rapid, flexible and secure deployments to large numbers of trading partners.
- Configurable Usage Policies – govern everything from supported vendor software, to required hours of availability, naming policies, error-handling techniques, and security requirements. Although some large enterprises develop and manage policies centrally through architecture groups, others choose to share policies through a distributed departmental structure. The important common thread is the existence of enterprise-wide policies and the ability of a technology to support those policies.
- Dynamic Configuration – provides a flexible technical environment to implement and modify policies, interfaces, business objects and services without impacting existing deployments.
- Integrated Security – The ability to determine variable access rights based on a user identity or class of identity. Security should also address data transport to ensure data integrity, confidentiality, authentication and non-repudiation.
- Reliability – Once a business process or object is placed into production, mechanisms should exist to ensure reliable access to that resource. In the case of data transport, reliability should address guaranteed once and once only delivery of data.
- Consolidated View – Given the broad range of potential participants in a Web services implementation, it will be increasingly important to provide a consolidated view, access rights permitting, that delivers relevant configuration and operational details for ALL participants and components.

## **Web services As Feature vs. Web services As Paradigm Shift**

The second comparison addresses the confusing array of vendor announcements and roles relating to the Web services ecosystem. Given the phenomenal interest in Web services, it is hardly surprising to find vendors from seemingly unrelated market segments scurrying to claim Web services support for their offerings. Most vendors are market driven and as such, try to keep pace with new technologies. This usually results in adding new "features" to their existing product line in an attempt to capitalize on the new technology buzz. These features are typically "bolt-ons", "wrappers" or "extensions" that comply with some portion of the new architecture

but do not represent a new way of approaching the problem. A paradigm shift, by contrast, usually represents a fundamental change to the architecture, assumptions and requirements of the problem domain.

Paradigm shifts in the software industry are usually accompanied by new vendors with solutions representing a fundamentally better approach to solving a problem or addressing a much broader set of requirements that their unique positioning uncovers. The challenge for end-users is to understand the Web services ecosystem and architecture well enough to distinguish between those vendors supplying Web services “features” and those with solutions built to capitalize on the paradigm shift.

## Adoption of Web services

Deploying something that meets the basic criteria of a Web service is almost trivial using today's development tools. Whether using Microsoft's .NET platform, IBM, BEA, an Enterprise Application Integration (EAI) product or one of the specialized Web services development toolkits flooding the market, generating WSDL, publishing to the UDDI registry of your choice and exposing via HTTP has almost been reduced to running a set of wizards. In many cases, Web services capabilities are no-cost or low-cost add-ons to existing tools further reducing the cost of implementation. This is positive to the extent that it promotes adoption of Web services, but ultimately leads to short-term inefficiency and chaos in most enterprises. This early stage of new technology adoption is based on the use of Web services as a feature and not Web services as paradigm shift. The move to enterprise-class Web services and the attendant benefits will coincide with the adoption of Web services as a paradigm shift and the ability to address the broader requirements of enterprise-class deployments.

**To illustrate this evolution from departmental deployments to enterprise-class deployments, it is useful to study the history and evolution of the EAI market:**

- The Early Days (1996-97) – The early days of EAI were characterized by incomplete vendor offerings, uncoordinated department-level implementations, many prototypes and a few critical point interfaces running but requiring constant attention. A common EAI implementation of this era might consist of a handful of point interfaces between an ERP system and key legacy applications. Vendors typically did one thing very well (i.e. transformation, messaging) and the broader community of analysts, press and users had yet to define the ideal EAI product stack. ROI was usually measured against a single project's goals without considering the broader impact on the organization.
- The Terrible Teens (1998-99) – This stage of the EAI market witnessed a number of key evolutionary steps. The vendors themselves began to evolve their technical offerings in response to broader user requirements, greater competition, increased exposure to the press and industry analysts and a drive to differentiate against a rapidly evolving view of the core EAI infrastructure. Users began moving interfaces from prototype to production and quickly learned how difficult it was to effectively manage their increasingly complex implementations. Most large enterprises took the logical next step of classifying

EAI services as enterprise infrastructure and assigning resources to develop enterprise-wide policies, procedures and architectures. ROI expectations rose dramatically as EAI vendors began selling their wares as critical enterprise infrastructure.

- The Glory Years (2000-01) – The latest stage of the EAI market really marks the beginning of enterprise-class products and deployments. Industry consolidation and shakeout have resulted in fewer vendors with reasonably complete enterprise-class offerings. The EAI community shares a common set of expectations with regard to features, function, scalability, operational support and architecture. Most enterprises are deploying second or third generation interfaces and architectures based on lessons learned and the natural maturity of the vendor offerings. The majority of new EAI purchases can be classified as enterprise decisions with input from architecture, policy and end-user groups. The market appears to be on the verge of achieving true infrastructure status where ROI calculations have more to do with the degree of EAI deployment and less to do with the decision to deploy EAI or not.
- The Future – The EAI market will continue to consolidate as major players either purchase or eliminate the weaker vendors. Standards such as the new Java Connector Architecture (JCA), Web services and XML should gradually simplify the integration challenge as more applications are built or modified to play by a common set of rules. EAI vendors will increasingly cast themselves as Web services enablers and compete head to head with the Application Server (AppServer) vendors.

This evolutionary path is not unique to EAI. Another recent example that is closely related to Web services is the Application Server market. The AppServer market follows a similar timeline but differs in how quickly standards impacted industry consolidation and enterprise adoption.

- The Early Days (1996-97) – Early app server vendors, responding primarily to the phenomenal growth in Web-driven e-business initiatives, began building new application hosting platforms that promised to simplify the effort while improving scalability. Many of the AppServer entrants chose to bet their future on the then new Java programming language. While providing many of the same features, almost every vendor's implementation was different thus producing high switching costs for end-users. The interesting by-product of the early stage was the rapid realization that certain core features like caching, connection pooling and distributed object support were common to every product and implementation.
- The Terrible Teens (1998-99) – Two major evolutionary steps took place during this period. IBM and BEA acquired AppServer technology vendors and began to apply their marketing muscle, effectively legitimizing this product category. Sun introduced Enterprise Java Bean (EJB) technology incorporating many of the common AppServer features into the core language specification. Given the large number of Java implementations, this presented an opportunity to standardize these common features by building support into the Java language specification while allowing the App Server vendors to focus on new features.

In reality, the rapid adoption of the EJB standards by vendors (BEA and IBM) and users helped to accelerate industry consolidation even more. As customers recognized the value of standards-based implementations, their confidence allowed them to move to enterprise-class implementations and purchasing decisions much more rapidly than in the EAI market.

- The Glory Years (2000-01) – Due in large part to the early consolidation of vendors and rapid adoption of standards, the AppServer market reached maturity much faster than the EAI market. The market quickly settled on 2 major vendors (BEA and IBM) and those vendors were successful in moving AppServer into the “must have” infrastructure category very quickly. As a result, the AppServer market is now focused on incorporating peripheral technologies such as EAI and Web services to sustain growth. The introduction of EAI related standards such as Java Messaging Service (JMS) and Java Connector Architecture (JCA) and the advent of Web services will likely alter the playing field considerably in the next few years.

Interestingly, Web services adoption is perceived to be accelerated by the fact that the leading EAI and App Server vendors have added support for Web services, meaning that their customers need not invest in an entirely new platform to fully embrace Web services. This perception is partially correct, because in fact Web services are nothing more than another feature on the long list that EAI and App Server vendors support. However, the key remaining hurdle will be to identify all of the requirements necessary to support Web services at the enterprise level and understand who will likely provide these services. Ultimately, to realize the full potential of Web services and support the predicted paradigm shift in software development, Web services must quickly mature to enterprise-class Web services with all their attendant benefits, some of which are not, and cannot be, delivered by a platform or tools. The goal of this white paper is to apply lessons learned from the EAI and App Server markets to identify the full Web services architecture needed to support enterprise class deployments.

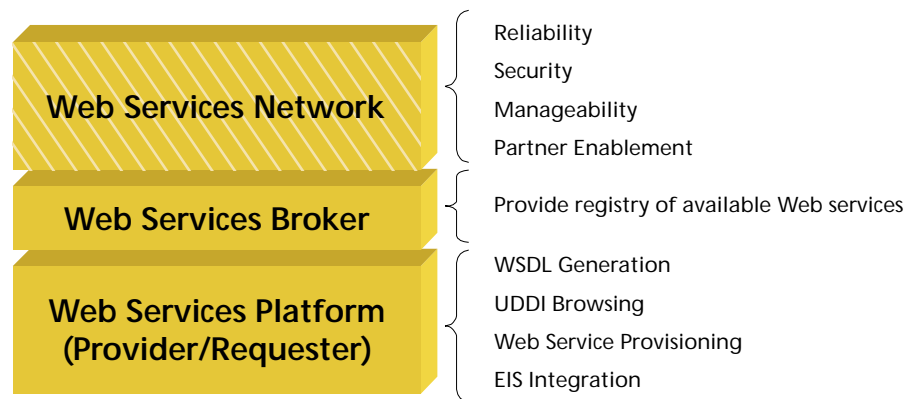
## Enterprise Web services Defined

In the previous section, a distinction was made between Web services and enterprise Web services. That distinction is largely a function of the broader requirements that need to be met as Web services move from ad-hoc prototype status to full-blown enterprise infrastructure. To recap, those additional requirements include:

- Physical Scalability
- Developmental Scalability
- Management Scalability
- Rollout Scalability
- Configurable Usage Policies
- Dynamic Configuration
- Integrated Security
- Reliability
- Consolidated View

Adding these enterprise-class requirements to the existing Web services architecture stack provides an interesting glimpse into the future of Web services.

## Enterprise-class Web Services



The remainder of this section will discuss each section of the Web services architecture stack in more detail with particular emphasis on how well that section meets the requirements for enterprise-class Web services.

### Web services Platform (Provider and Requester)

The early definitions of the Web services architecture typically included two mandatory components:

- Service Provider or Publisher – The provider is generally responsible for:
  - o Providing the business logic
  - o Generating a WSDL description of the interface(s)
  - o Providing the HTTP invocation mechanism
  - o Providing marshalling between SOAP and the business object or XML document
  - o Publishing to a UDDI-compliant registry

Depending on the product you choose to use, the provider may also include an application server to host the business logic and EAI capabilities to integrate with existing applications and data sources.

- Service Requester or Client – The consumer of the provider's service, the Requestor is generally expected to support:
  - o Browsing of UDDI compliant registries
  - o Parsing WSDL documents and auto-generation of code stubs for implementation
  - o Invoking published URI using HTTP/HTTPS at a minimum
  - o Automated marshalling between SOAP and the business object or XML document

Depending on the product chosen for implementation, the requester platform may also provide additional capabilities such as EAI capabilities.

Given the recent explosion in interest in all things Web services, it shouldn't be surprising that virtually every software vendor claims some level of Web services compliance. Nevertheless, for our purposes, we'll concentrate on a few major vendor categories that focus specifically on providing Service Publisher and Requestor tools.

- **Development Tools** – Almost all of the development tool vendors (including AppServer vendors) have released or announced support for the core provider and requester functions listed above. Examples include IBM, Microsoft, BEA, Visual Café, Jbuilder, Cape Clear and many others. The vendor offerings may differ in ease-of-use and maturity, but they all strive to meet the basic requirements.
- **EAI Vendors** – Most of the major EAI vendors (e.g. Tibco, Vitria, WebMethods, SeeBeyond and Mercator) have announced support for both Provider and Requestor functions.
- **Application Vendors** – Most of the major application vendors for ERP, CRM, SCM (e.g. SAP, PeopleSoft, Siebel and Oracle) and other popular packaged applications have announced a Web services strategy or direction. Typically for an application provider, Web services support means they will expose their business objects and processes as Web services in some future release.
- **Portals** – The concept of a portal vendor is fairly vague these days given the increasingly generic use of the word to describe a consolidated view of and access to content regardless of the source. In fact, most of the vendors in the three prior categories provide portal tools or solutions at some level in addition to a list of pure portal technology vendors (e.g. Epicentric, Plumtree). The importance of portals has more to do with their emerging role in the Web services ecosystem. Many enterprises view Web services as a mechanism to extend portal content and functionality directly to applications. In other words, the portal's role will be to aggregate content access from a variety of sources and make it available for both human and application-to-application access.

## Web Services Broker

A Web Service Broker, or directory, plays the optional role of matchmaker in the enterprise Web services architecture. The broker function, while not strictly required, is widely assumed to be a critical component to facilitate dynamic discovery of desired Web services much like the yellow pages of today. While this is certainly a necessary function for building enterprise-class Web services that scale to support a large number of external, public partners, it is also a critical component when deploying Web services within an enterprise or for use with private trading communities. In fact, the distinction between public and private directories already exists for most implementations with early rollouts favoring the reduced scope and manageability of a private directory. While IBM, Sun and Microsoft are to be credited with providing early UDDI compliant implementations for public consumption, it appears that the vast majority of early Web services deployments are utilizing private brokers often hosted within an enterprise.

The primary requirement for a Web Service Broker is to implement the latest UDDI specification for publishing Web service-related data to a community of users. To implement Web services without a Service Broker, users would exchange the WSDL document and service URI directly with a chosen partner. While this works fine for a small number of deployments with known partners, it lacks the scalability and dynamic nature of directory-based implementations.

So what's missing? When the roles of Service Requester and Service Provider are mapped against the requirements of an enterprise-class Web services architecture, the gaps in functionality shown below emerge:

Requirement	Service Provider	Service Requester
Physical Scalability	Depends on development platform	Depends on development platform
Developmental Scalability	Depends on development platform	Depends on development platform
Management Scalability	User must implement	User must implement
Rollout Scalability	Limited support	Limited support
Configurable Usage Policies	User must implement	User must implement
Dynamic Configuration	Depends on platform	Depends on platform/user must implement
Integrated Security	Usually exposes server platform security features	User must adhere to Provider security scheme
Reliability	Limited to platform availability. No measure of service availability across the network.	User must implement.
Consolidated View	Lacks view of all participants/components and therefore cannot address.	Lacks view of all participants/components and therefore cannot address.

The functionality gaps, critical to the definition of enterprise-class Web services, are being addressed by the third component: Web Services Networks.

## Web Services Networks

Web Services Networks are a relatively new component of the enterprise Web services architecture. They were created in response to the enterprise functionality gaps identified in the previous section. Conceptually, they provide a higher-level view of the Web services components that are needed to provide key capabilities in the areas of reliability, management/monitoring and security.

### **A simple example illustrates the value of a Web Services Network:**

Widgets, Inc. provides a Get\_Order\_Status service to their customers. It was designed to provide computer-to-computer access to a customer's SCM software. Widgets deployed their Get\_Order\_Status Web service using IBM's Websphere platform. Using tools provided by IBM, Widgets monitors the Web service to ensure 24x7 availability.

ACME Corp. is a customer of Widgets, Inc. They rely on Widget's Get\_Order\_Status Web service to keep their customers informed of delivery dates. ACME used Microsoft's .Net platform to build a Web application for their customers that uses an ACME order number provided by the customer to look up all the corresponding outsourced part order numbers then retrieves their status using Web services. The results are run through the appropriate SCM modules to produce a projected completion date.

True to their word, Widget's application is available 24x7. Unfortunately, and unknown to Widget's, the timeout parameter on their Web service was set too low to accommodate the increased network load that occurred each day at 8am, noon and 5pm. As a result, about half of ACME's Web service requests were timing out resulting in inaccurate information to their customers.

Now run the same use-case using a Web Services Network. The Web Services Network has been configured to support ACME's access to Widget's Get\_Order\_Status Web service. Every time ACME attempts to invoke Get\_Order\_Status, the Web Services Network logs key information, including success/failure and makes it available to the appropriate administrator at Widget's. The Widget's administrator configured the Web services Network to send an email alert when certain availability thresholds were not met and was able to pinpoint and correct the problem very quickly.

There are a limited number of options for providing the Web services network functionality. The three most common are described below and analyzed for effectiveness in meeting the requirements of enterprise-class Web services.

#### ***Networked EAI Trading Partner Servers***

Much has been written about the evolving role of the EAI vendor in the new Web services ecosystem. Most of the EAI vendors have now announced eventual support for the core Web services standards with the goal of supporting both Service Provider and Service Requester functionality. They are obviously in a position to accelerate the adoption of Web services short-term by providing a bridge or wrapper to existing systems and data until such time as those systems can provide native Web service capabilities.

What is less clear is the ongoing role of the EAI vendor as Web services become part of the standard Enterprise Information System (EIS) infrastructure. As Web services begin to permeate the integration realm, proprietary messaging, adapters, proprietary business process integration and transformation (a significant portion of the EAI product stack) become less and less relevant.

One possible scenario finds the EAI vendors repurposing their necessarily complex, feature-laden, integration servers as lightweight Web Service Network agents or proxies by disabling, removing or de-emphasizing vestigial capabilities. An early example of this repurposing effort was the EAI community's largely ineffective attempt to market "Onramps" or "Spokes" to the large public exchanges whose business models were dependent on accelerating transaction volume. These Onramps/Spokes were typically lower-priced, license-restricted versions of the EAI vendor's full integration server platform with packaged connectivity templates for the sponsoring exchange. Unfortunately, due to the relatively sudden demise of the public exchange market, it is unclear how effective these repurposed EAI integration servers were as lightweight enablers. What is clear is that the recent emergence of Web services and Web Services Networks will forever change the landscape of partner enablement by offering purpose-specific, optimized solutions that are fundamentally better suited to the task of facilitating inter-company business process sharing.

### ***Hub and Spoke "Value Added" Networks***

The second variation of a Web Services Network is the direct descendent of the Internet Value-Added Network which is in turn a descendent of the original Value-Added Networks associated with EDI. Example providers include GE, Sterling Commerce, Peregrine, Viacore, and Grand Central. Value-added networks or VANS were originally built to support the store-and-forward nature of EDI documents in the pre-Internet era where device and protocol compatibility were the major obstacles. By building a backbone of network access points and centralized mailboxing, archiving and security, VANS provided the communication infrastructure necessary to enable commercial exchange of business documents with a reasonable level of quality and reliability.

The explosion of dedicated Internet access in the 90's signaled the end of the now outdated asynchronous dial-up networks that represented most VANS. Most large EDI customers were moving to FTP over existing high-speed lines and were demanding flat rate pricing vs. the traditional "per kilocharacter" models widely used by most VANS. The emergence of TCP/IP-based connectivity significantly lowered the barriers to entry in the VAN marketplace and a number of new Internet VANS were born. These new VANS still provided the traditional hub functions of the EDI VANS but employed business models that took advantage of cheaper, standards-based infrastructure and were better suited to the emergence of XML as the eventual successor to EDI.

Fortunately, or unfortunately, the demise of EDI was greatly exaggerated and the old EDI VANS were reasonably responsive to the changes in technology. By the time the Internet VANS began to get their message out to the market, the technology had morphed again to Web services. As a result, the so-called Internet VANS are currently repositioning their offerings as Web services Networks but still employing the traditional hub and spoke architecture. While the hub and spoke approach still addresses many of the enterprise Web services requirements, there are two significant technical limitations that will limit their viability:

1. Sub-optimal support for Remote Procedure Call (RPC) style (or synchronous) Web services – RPC-style Web services are an important part of the Web services use case mix. They represent a significant percentage of likely Web services deployments and are characterized by the need for immediate response. While many will argue that loosely coupled asynchronous designs are better in the long-run, they are significantly more difficult to design and deploy and often represent the first trade-off in projects, particularly when staffed by less-than-expert developers or developers moving from procedural to object-oriented design. As a result, RPC-style Web services are likely to represent a significant proportion of early Web services deployments. Forcing RPC-style Web services through a central hub adds overhead and potential failure points increasing the complexity of the deployment.
2. Lack of suitability to both Intra and Inter-enterprise Web services – It is reasonable to expect that most initial Web services deployments will happen inside the enterprise where users have greater control over the security and consequences of new technology deployments. It is unlikely that enterprise users will allow their internal data and business processes to flow outside their protected confines, through the hub and spoke Web Services Network then back into the enterprise. This adds unnecessary overhead, latency and risk to an intra-enterprises business process.

### ***Native Web Services Networks***

The third category of Web Services Network emerged in direct response to the requirements for enterprise Web services and as such was not constrained by a repurposed or legacy architecture. It employs a peer-to-peer architecture with lightweight proxy or agent providing optimized routing, monitoring and management of both synchronous and asynchronous Web services with a centralized management system. Equally suited to both intra and inter-enterprise deployments, this class of Web Services Network was designed from the ground up to meet the requirements of enterprise class Web services. Flamenco Networks is the first provider of a native Web Services Network.

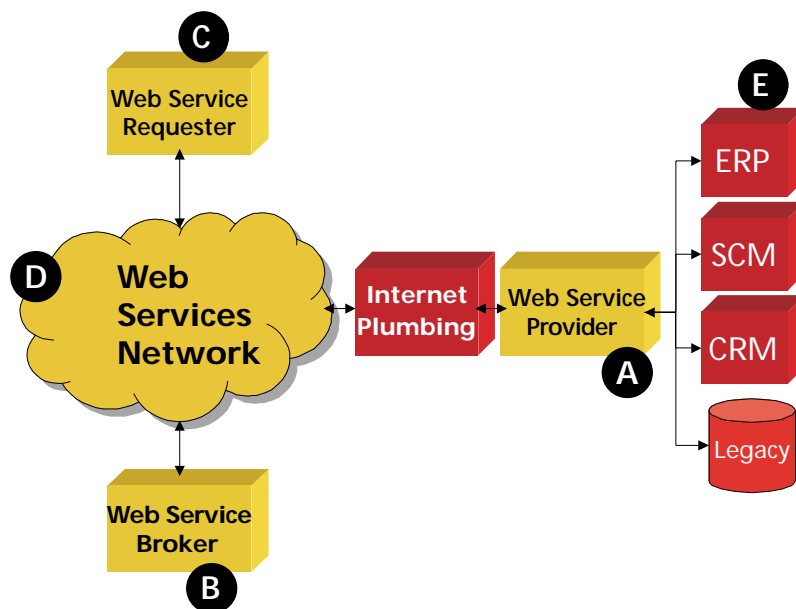
The table below compares the three different types of Web Services Networks to the previously documented requirements for enterprise Web services:

Web Services Network Requirement	Networked EAI Trading Partner Servers	Hub and Spoke "Value Added" Networks	Native Web Services Networks
Physical Scalability	Varies by EAI vendor, but typically optimized for large number of protocols and use cases trading scalability for versatility. Each node typically limited to communications with a single partner.	Limited by need to route all Web services (inter and intra- enterprise) through a central hub.	Optimized for Web services protocols and standards, with particular emphasis on XML, HTTP and SOAP parsing.
Developmental Scalability	Varies by EAI vendor, but typically optimized for multiple internal development teams using wide range of protocols, data formats and adapters.	Issues surface when moving intra-enterprise deployment to inter-enterprise deployment.	Optimized for ease-of-deployment, co-existence with Service Provider and Service Requester tools and security requirements. Migrating from intra to inter-enterprise deployments is simple administrative change.
Management Scalability	Designed to accommodate wide variety of protocols and data formats, minimal emphasis on multi-partner monitoring.	Lack of suitability to RPC-style Web services and combination of intra and inter-enterprise requirements may cause problems.	Symmetric design, optimized for Web service protocols and use cases, provides rich management and monitoring of Web service interactions both intra and inter-enterprise.
Rollout Scalability	Varies by EAI vendor, but most most rollouts hampered by lack of standards, complex implementations and ill-defined use cases.	Hub and spoke architecture places heavier burden on participants requiring API-level invocation	Integrated applications that automate rollout of WSDL to massive numbers of partners with self-service provisioning and URL invocation rather than proprietary APIs
Configurable Usage Policies	Varies by EAI vendor, but most have not designed Web services-specific policy architectures and therefore most employ more complicated mechanism	Based on manual configuration files or hard-coding.	All configuration managed centrally via Web-based configuration and control facility using declarative rules.
Dynamic Configuration	Varies by EAI vendor's integration server capabilities.	Reliance on SDK and configuration files limits this capability.	Centrally managed configuration and control interacts with proxy to effect dynamic configuration.
Integrated Security	Varies by EAI vendor and enterprise requirements.	Integrated, configurable.	Integrated, configurable.
Reliability	Varies by EAI vendor and protocol being deployed.	Supports message retries, tracking, timeouts and duplicate avoidance mechanism.	Supports timeouts, resends, guaranteed once and once only delivery and tracking.
Consolidated View	Would require EAI vendor software to be installed/used at each participant location.	Available for inter-enterprise but probably not available for intra-enterprise deployments given previously cited issues regarding overhead and security.	Provided for both inter and intra-enterprise deployments.

Based on early market feedback, it is important to re-emphasize the importance of providing support for a seamless transition from intra-enterprise to inter-enterprise deployments. It appears that many enterprises are choosing intra-enterprise deployments as their initial foray into the world of Web services for the usual reasons related to security, resource control, and minimization of business disruptions with external partners. Simple, declarative configuration rules coupled with the lightweight footprint requirements of a native Web Services Network provide an ideal foundation for those enterprises choosing to migrate from tightly controlled internal deployments to widely available public deployments with minimum effort and maximum control.

## The Enterprise Web services Ecosystem

Now that we've defined the enterprise Web services architecture in more detail, it may be useful to examine the enterprise Web services ecosystem and a few Web services uses cases to further illustrate the concept. The diagram below identifies the usual functional participants in any Web services use case.



The Web services Ecosystem consists of the functional components needed to deploy Web services, the common use case variables, and the vendor community providing the tools. The next section will describe the common variables found in Web services use cases. The final section will provide detailed use cases and how they benefit from an enterprise Web services implementation.

## Examples of Web Services Use Case Variables

Web services can typically be classified in two ways:

### *RPC-Style Web services vs. Message-Style Web services*

RPC-style Web services are typically characterized as synchronous, relatively simple parameter structure values typically returned as name/value pairing with no complex XML parsing required. An example might be the ORDER\_STATUS method where order number is the IN parameter and status is OUT parameter. RPC-Style Web services are typically used directly by the person writing business logic vs. a generic pub/sub architecture where documents are routed based on content. In practice, using RPC-Style Web services may require a sequence of Web service invocations, but the duration of the entire sequence is typically short.

Message-style Web services are characterized as asynchronous with the potential for highly complex document structures (including CDATA sections containing non-XML content). Message-Style Web services typically require transformation and routing based on content and are frequently part of longer running sequences. An example would be an order/order acknowledgment Web service pair requiring the transmission of EDI-like quantities of data.

### *Intra vs. Inter-enterprise Web services*

The distinction between inter and intra-enterprise is not always clear. Intra means within while Inter means between or among. In the integration world, the distinction is usually based on a combination of legal entities involved and technical boundaries. For the sake of discussion, we'll assume inter-enterprise Web services are between two unrelated businesses with separate physical networks separated by numerous firewalls and the public Internet.

Although it is common to associate Message-style Web services with inter-enterprise use cases, this is not technically accurate. Any combination of RPC-style and Message-style Web services may be used in any sequence both intra and inter-enterprise. In fact, one of the benefits of using a Web Services Network is the ability to reliably deploy and manage composite Web services without the relying on complex protocols such as RosettaNet.

## Use Cases

This section will describe two Web services use cases, how they might evolve over time and the short and long-term benefits of deploying within the enterprise Web services ecosystem including the use of a Web services Network.

### *Order Status Portal*

In this use case, Web services are chosen as the deployment mechanism for ACME's new customer service portal designed to provide self-service for order status checking. The new portal is being developed to run on a leading App Server using their latest Web services tools. Initially, the portal will provide access to the aggregate ORDER\_STATUS function in the company ERP system. Over time, it is envisioned that customers will be given the option to check status on individual components of the order even when those components are being provided by a third-party supplier. The ORDER\_STATUS method will support both browser-based interaction and application-to-application interaction using Web services.

#### **Initial Functional Architecture**

The ORDER\_STATUS Web service will utilize an existing function in ACME's ERP system that accepts an order number and returns a status message. The initial ORDER\_STATUS Web service will require ACME to create new business logic on their chosen App Server that will:

- Accept and parse the SOAP envelope, optional headers and body
- Formulate a function call to the ERP system using the correct syntax and protocol
- Accept and parse the response from the ERP system
- Package the response back into an appropriate SOAP response object and return to the initiating application

It is further assumed that ACME will deploy the ORDER\_STATUS Web service using the servlet engine provided by the App Server vendor and a commercial quality Web Server. Basic error-handling functionality will be provided using SOAP FAULT elements.

The ORDER\_STATUS Web service description will be made available in WSDL format in a private UDDI-compliant directory maintained by ACME. ACME's Web Services Network will also host a copy of the ORDER\_STATUS WSDL document on their UDDI-compliant directory. ACME will provide access to the ORDER\_STATUS Web service exclusively through their native Web Services Network.

#### **Anticipated Changes**

The ACME ORDER\_STATUS method provides an aggregate view of the ACME order fulfillment process, but only at the rollup level. Many ACME products are composed of parts and sub-assemblies from non-ACME suppliers. ACME would like to extend their ORDER\_STATUS Web service to provide their customers with greater detail on the exact status of their orders by providing status at the component level. ACME believes that by providing better visibility to their customers, they will improve customer loyalty and reduce the number of manual inquiries handled by their service department.

The technical changes required to provide this detailed ORDER\_STATUS Web service include:

- Convincing each ACME supplier to expose the ORDER\_STATUS of their components as a Web service
- Implementing the sequence of ERP function calls that retrieve appropriate vendor information for each part of an ACME order
- Formulating the Web service Requester logic to retrieve the ORDER\_STATUS for each component/sub-assembly by vendor
- Formulating the Web service provider logic to reassemble the individual responses into a single, aggregate Web service response to ACME's customer

Once these changes are in place, ACME will publish a new WSDL document describing the ORDER\_STATUS Web service with its richer detail. Customers would have the option of using either the new or the old ORDER\_STATUS based on their unique needs.

### Benefits Of Deploying Enterprise Web services With Web Services Network

ACME chose to deploy the ORDER\_STATUS Web Service through a Native Web Services Network to address their requirement to provide enterprise-class Web services to their customers. The table below uses the enterprise-class Web services requirements defined earlier in this paper to highlight the benefits of deploying ACME's ORDER\_STATUS Web Service through a native Web Services Network.

Enterprise Web Service Requirement	Native Web services Network Benefit
Physical Scalability	Decentralized architecture of native Web Services Network eliminates processing and network bandwidth bottlenecks. Web Services Network scales to handle initial and extended ORDER_STATUS deployments.
Developmental Scalability	Less of an issue for the initial deployment, the extended deployment supports independent development, testing and deployment by each participant in composite ORDER_STATUS Web service. Also supports team-based development, access control and configuration during build-out of composite Web service.
Management Scalability	Provides robust, granular monitoring and management for composite Web service allowing ACME to predict and troubleshoot any individual component, even when involving supplier Web services.
Rollout Scalability	Provides tools and platform to accelerate access to unlimited number of ACME customers without requiring internal investment.
Configurable Usage Policies	Provides simple, declarative mechanism for migrating from test to production and intra-company to inter-company. Eliminates dependence on "hard-coding" and maintenance issues that follow.
Dynamic Configuration	Supports separation of business logic from deployment details. Allow developers to focus on core task without regard to variability of vendor composition, user community or scope of use case.
Integrated Security	Provides robust, consistent security regardless of user or supplier platform. Significantly reduces complexity and cost of Public Key Infrastructure (PKI) implementation for Web services.

Enterprise Web Service Requirement	Native Web services Network Benefit
Reliability	Built-in support for guaranteed delivery, retries and configurable timeouts simplifies development effort for ACME staff while providing reliable, manageable implementation.
Consolidated View	Single view of all components (both ACME supplied and user/supplier supplied) supports proactive management of service levels even when deploying complex, multi-part Web services.

The benefits derived from the use of a native Web Services Network are clear and compelling. It should also be noted that many of the benefits would be extremely difficult to achieve without the unique architecture of the native Web Services Network. And while anything is possible, the technical complexity, cross-company coordination and need for standard policies and procedures would make a comparable ACME developed solution cost and time prohibitive.

### *Supply Chain Automation*

In this use case, Web services are chosen to automate the process of sourcing and managing sub-assemblies for a moderately complex product. The initial implementation calls for a combination of RPC-style Web services (to check pricing and availability) and Message-style Web services (to exchange order, acknowledgment and shipping advisory data in the form of XML documents) between ACME and their top 3 suppliers. Over time, it is anticipated that ACME will extend this process to approximately 50 potential suppliers.

### **Initial Functional Architecture**

Much like the ORDER\_STATUS use case, ACME's Supply Chain Automation (SCA) initiative will be deployed as a new composite application hosted on an Application Server. Much of the supply chain business data, logic and process are implemented in ACME's ERP system with the ERP system acting as a sort of data and process warehouse to the new Supply Chain Automation initiative.

A typical order process might entail the following steps:

1. An order is placed via one of ACME's many ordering mechanisms (i.e. EDI, Fax, Call Center, XML document, browser).
2. The order is submitted to the ERP system.
3. The ERP system initiates a process to determine the composition, vendor, pricing, scheduling and delivery of the components and sub-assemblies needed to fulfill the order.
4. Any step in the process requiring real-time data would call the appropriate method in the SCA application, such as GET\_CURRENT\_PRICE, passing the required parameters.
5. The SCA application would formulate the ERP system call into a Web service request after retrieving the appropriate data from the initiating request and possible subsequent queries to the ERP system for related data.

6. The SCA application would initiate the Web Service request to one or more vendors and return the best alternative to the ERP system as a response to the initial GET\_CURRENT\_PRICE call.
7. The ERP system would continue the order process until it was ready to submit orders for the various components and sub-assemblies needed to complete the product.
8. After selecting the vendors, the ERP system would publish the individual orders for each component/sub-assembly as EDI documents, XML documents or proprietary formats.
9. The SCA application, subscribing to these events, would convert the ERP system order document if necessary, and then package into a SOAP message and Post to the vendor supplied PLACE\_ORDER Web service.
10. Given the likely need for time to process and check their own supply chain systems, the vendor would probably be expected to return a simple acknowledgment with a full response document such as a PO Acknowledgment to be sent at some time in the future.
11. ACME would provide an ORDER\_ACK Web service to receive these asynchronous responses at some later time, ultimately returning the data to the ERP system to complete the order process loop.

This represents a fairly complex process in that it involves a combination of RPC-style and Message-style Web services in a somewhat extend period of time and requires coordination with multiple third parties. It should also be noted that the decision to implement the SCA initiative as a new composite application on an Application Server was due to the absence of equivalent functionality in the current release of the ERP implementation. This entire process could just as easily have been provided as an integrated feature of the ERP system and probably will be in the future as ERP vendors incorporate Web services into their core architecture.

### **Anticipated Changes**

The SCA application is fairly ambitious in initial scope so most of the anticipated changes are related to number of participating vendors. The initial implementation will be limited to three trusted vendors to minimize system disruption. Once the process is tested and refined, the number of vendors will be expanded to 50. The final stage of implementation intends to support dynamic acquisition of vendors through a registration Web service made available to all vendors.

One other possible change is the eventual migration of the SCA application functionality from the new composite application back into the ERP system. ACME's ERP vendor has already announced support for Web services in a future release and ACME intends to migrate to that platform as it becomes available.

## Benefits Of Deploying Enterprise Web services With Web Services Network

Once again, the benefits of using a native Web Service Network are best described using the enterprise-class Web services criteria discussed previously. Clearly, the scale of a Web services initiative tends to amplify the benefits of a native Web Services Network given the increased demands in almost all aspects of the deployment.

Enterprise Web Service Requirement	Native Web services Network Benefit
Physical Scalability	Decentralized architecture of native Web Services Network eliminates processing and network bandwidth bottlenecks. The Web Services Network is designed for the large number of partners anticipated in future stages of the SCA initiative.
Developmental Scalability	The relatively complex nature of the SCA initiative probably dictates multiple development teams throughout the deployment cycle. The native Web Services Network supports independent development, testing and deployment both within ACME and among the vendor community.
Management Scalability	Provides robust, granular monitoring and management for composite Web service allowing ACME to predict and troubleshoot any individual component, even when involving supplier Web services. This is particularly important when deploying a composite Web service with both RPC-style and Message-style implementations.
Rollout Scalability	Provides tools and platform to accelerate access to unlimited number of ACME customers without requiring internal investment. Moving from 3 to 50 to 500 trading partners will be transparent to ACME.
Configurable Usage Policies	Provides simple, declarative mechanism for migrating from test to production and intra-company to inter-company. Eliminates dependence on “hard-coding” and maintenance issues that follow. This benefit applies even if ACME migrates the complete SCA application from the App Server deployment to a native ERP deployment. From the Web Services Network point of view, this represents a simple configuration change.
Dynamic Configuration	Supports separation of business logic from deployment details. Allows a developer to focus on core task without regard to variability of vendor composition, user community or scope of use case. This is particularly relevant when migrating deployments from one platform to another.
Integrated Security	Provides robust, consistent security regardless of user or supplier platform. Significantly reduces complexity and cost of Public Key Infrastructure (PKI) implementation for Web services. This benefit accelerates significantly as the number of partners increases from 5 to 500.
Reliability	Built-in support for guaranteed delivery, retries and configurable timeouts simplifies development effort for ACME staff while providing reliable, manageable implementation. Consistent applicability to both RPC-style and Message-style Web services saves development time, maintenance and ongoing management cost.
Consolidated View	Single view of all components (both ACME supplied and user/supplier supplied) supports proactive management of service levels even when deploying complex, multi-part Web services. This is of particular importance when deploying complex, composite Web service systems such as the SCA application.

## Summary

This white paper discusses several important trends in the rapidly evolving Web services market:

- The rapid adoption and evolution of the Web services ecosystem provides ample proof that Web services have arrived and are here to stay.
- Most new technologies evolve to support the shift in implementation patterns from departmental to enterprise-class. A brief review of the history of the EAI and App Server markets lends credence to this pattern and portends a similar but accelerated shift in the rapidly maturing Web services market.
- The requirements for an enterprise-class Web services architecture outstrip the capabilities of Web services development tools and deployment platforms alone. Native Web Services Networks play a crucial role in supporting the paradigm shift represented by Web services.

The use cases were provided to further explore the requirements of an enterprise-class Web services strategy while illustrating the benefits acquired through use of a native Web Services Network.