



## A ROADMAP TO BUILDING AN ESB



**Author**  
Kiran Kanetkar  
Middleware Practice Manager  
Sateri Systems, [kiran.kanetkar@saterisystems.com](mailto:kiran.kanetkar@saterisystems.com)



### Abstract

*Enterprise Service Bus (ESB) as concept has been catching a lot of attention in recent times. It is necessary to understand what ESB is supposed to deliver before any investment is made to build one. Building an ESB will involve many different players in an organization. It is necessary to bring all those players on same page on what to expect when you start to build.*

### Keywords

SOA  
ESB  
UDDI  
SOAP  
Governance

### ESB Definition

ESB is an infrastructure component that enables Service Oriented Architecture (SOA). A number of articles have been written about SOA being a way of designing applications. SOA strives to achieve loose coupling between communicating applications. In a service orientation, an application (Server side) is designed to service request from another application (Client side). The application servicing the request becomes the Service Provider whereas application making the request becomes the Service Consumer of the service. So in simple terms ESB provides communication pipe between Service Provider and Service consumer to communicate seamlessly. The pipe can be used to route many different types of requests. That's where standards come into picture. The host of web services standards defines how various interactions between service providers and consumers

should take place. You may have J2EE based service and .Net based client or vice versa but they should be able to communicate as long as they use standard protocols such as SOAP/HTTP or SOAP/JMS.

An enterprise typically implements many different applications. Some of them are packaged applications such as ERP, CRM etc. or they can be legacy based applications such as mainframe or they can also be custom developed applications based on J2EE standards or .Net framework. These applications will be at different stages of their life cycle and some may support web services standards readily whereas others might need some more work to be compliant with the standards. Given this kind of scenario, it is imperative that any organization will have a mixed environment consisting of web service ready applications and applications that need to be enabled for web services. An ESB plays a very important role here since it tries to make it transparent whether a consumer or a provider supports web services standards.

### ESB Functions

Let us a look at some of the functions an ESB should provide and what it takes to build that functionality

- Routing – A service provider offers a service at an endpoint. Consequently the service requests need to be routed from the consumer to the provider's end point.
- Transformation – A service request might need to be transformed from one format to another. It might be XML to XML transformation using XSLT.
- Adaptation – Messages in an ESB most of the times follow standard SOAP format. However some application may not support SOAP

format and might need some kind of adapter to transform the message.

- Messaging – Provides asynchronous reliable messaging transport
- Orchestration – An integration scenario might require state information to be managed by an orchestration engine that orchestrates the flow of control from one service to another.
- UDDI Registry – The services are registered in the registry for applications to discover them. A registry will be used at design time to find out details about the service using a WSDL file. It will also be used at run time to locate the valid endpoint for the service.
- Security – Authentication and authorization of entities making the service calls using ESB infrastructure
- Consumer Integration – Ability for consumers to lookup the endpoint and direct the service requests to that endpoint.

- Service Integration – Validating the service requests and enforcing policies to enable secure service invocation
- Metrics and Management – Monitoring the ESB infrastructure to measure the key performance variables.
- B2B – enable service interaction with external parties. It might involve invoking external services or allowing external parties to invoke internal services over a firewall.

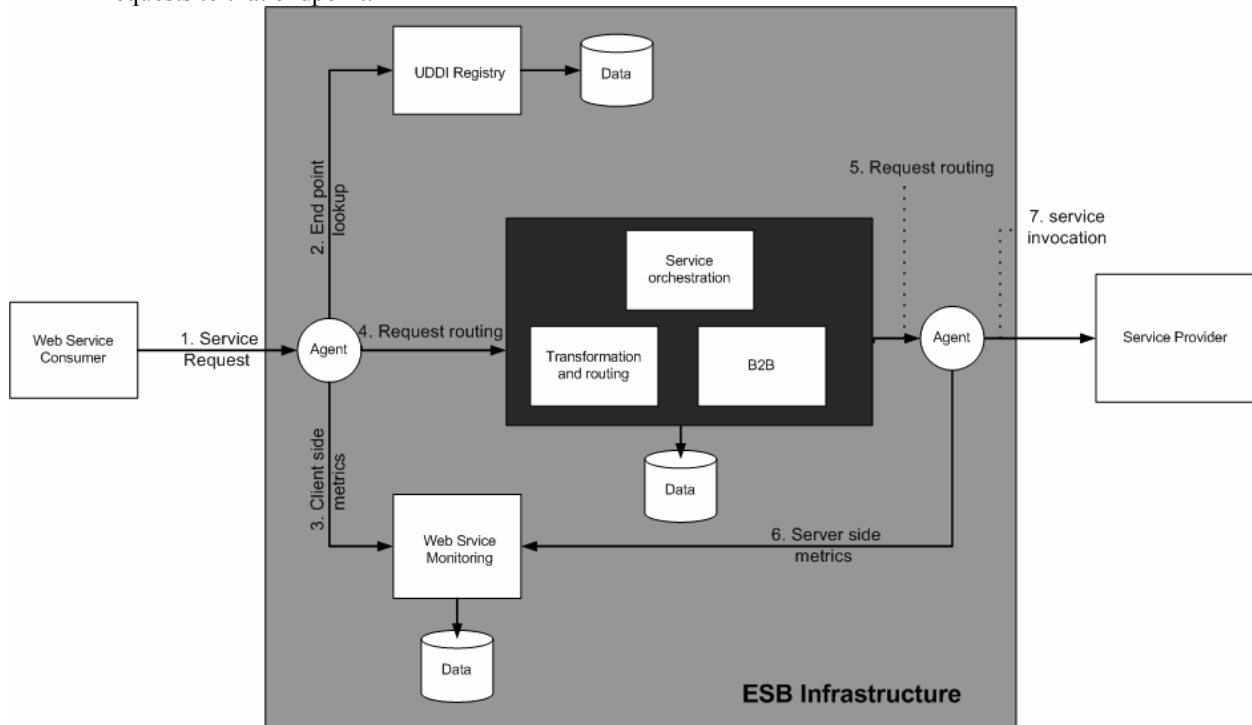


Fig 1 ESB Architecture

## Project Planning

The typical ESB implementation project will have to follow enterprise project methodologies such as Plan -> Build -> Operate. The key thing before planning is a proof-of-concept to do a product selection. The rest of the planning will depend on what products you choose for an ESB.

## Product selection

We discussed at high level functions true ESB should provide. An analysis of the functions will tell you that there is no single product available to do all this. Hence any organization thinking of building an ESB infrastructure will have to look at multiple vendor products to provide the various functions. The diagram

above does not include some of the base infrastructure such as LDAP servers, Single Sign-on and Identity management software that ESB will depend on for security. The product selection criteria can be devised based on the ESB functions and ability to implement them by different products. Actual organizational integration scenario that will be handled by ESB should be used to test the capabilities of different products.

### Technical Planning

Planning the functional deliverables is very important when it comes to effectively managing ESB deployment. A list of supported functions can be prepared in the form of Use cases that the ESB needs to support. Use cases identify all the actors involved in executing a particular function. This facilitates in gauging the scope of work for each function. The use cases can be of two types

- High Level use cases
- Detailed Use cases

High level use cases identify all the functions supported by ESB as discussed earlier. These can then be broken down into phases by which they will be delivered. For each phase a set of use cases to be delivered should be identified. This set of use cases should be then converted into detailed use cases. Detailed use cases identify the requirements for each use case in minute details. They also detail out interaction between different applications and / or people in step by step manner. The requirements identified for each detailed use case become the basis for design documents for each component. The design documents can then be translated into a high level architecture document that captures interaction between different components

### Deployment Architecture

The technical planning phase helps to identify detailed requirements for each component within the ESB. These requirements need to be translated into a System architecture diagram that needs to meet these requirements. For example requirements might identify that UDDI registry needs to be available on 24/7 basis. The infrastructure that supports such high availability needs to have clustering and failover capability. The relevant software products need to be configured to support clustering and failover. The architecture needs to support capability to upgrade software products without affecting normal operation. During initial phases a high level architecture diagram helps identify hardware requirements for ESB. The hardware specifications have to be finalized and hardware ordered well in advance to support timely deployment.

The hardware ordering consists of many different things such as

- Windows or Unix servers for the vendor products
- Storage disks with sizing considerations
- Power and racking requirements at the data centres
- Any additional hardware to support clustering and high availability

The detailed architecture showing how the different components interact with each other helps understand how those components need to be built. This architecture becomes the basis for build specification. All the build activities in right sequence can be planned to have timely deployment of all the components.

### Build and Test

The architecture document and hardware specifications identified during design phase need to be used to build the various environments. The built infrastructure can then be tested using functional tests, integration tests, performance tests and failover tests. Functional tests cover the ESB functionality for a particular component whereas integration tests should be across the components. These tests should cover all the use cases identified for a particular phase. Exit criteria for each set of tests can be set to ensure that the infrastructure is robust enough to satisfy the requirements set out for each use case.

### Challenges

When implementing enterprise wide infrastructure project there are challenges on multiple fronts.

- **Multiple Vendors** - When we talk of multiple vendor products to support an infrastructure the first thing that comes to mind is compatibilities between different products. When it comes to SOA it also means interoperability issues between J2EE and .Net based applications. Given the heterogeneous nature of different applications within an enterprise there will always be a mix of J2EE and .Net rather than having pure J2EE or pure .Net environment.

Managing a mix of products in an infrastructure can be very painful considering the fact that these products are at different stages of their life cycle. Hence you need to prioritize set of functionalities that should be delivered in a phased manner. The criteria for prioritization is a combination of what the current vendor product version supports and

what is deemed essential for your ESB infrastructure.

- **Organizational** - Challenges within the organization are also equally important in delivering effective ESB. An ESB infrastructure needs certain pre-requisite operational applications such as SSL certificate management, LDAP registries, identity management systems etc. The people managing these applications also need to understand what the ESB is supposed to deliver and what is required of them to enable that. Governance by it self can be a separate topic of discussion. Hence organizational readiness has to be checked before launching ESB deployment project. Various groups within the organization need to be brought on same page and their expectations need to be set. Without the organizational readiness assessment you might loose precious time in the middle of implementation if you have to bring up the different groups to speed on what to expect in building ESB infrastructure.
- **Standards** – Following standard is key to success when implementing SOA. While web Services standards is a must you also need to have standards for namespaces, XML schema, WSDL files, Business domains to name a few. The central organizational group should handle building of these standards and ensure that different IT users follow those standards. This group will also be responsible for establishing organizational best practices and guidelines for Service orientation.

## Migration to ESB

Many organizations are already using some sort of broker technology for their EAI needs. Typically such implementations follow a hub and spoke type of architecture with broker at the hub and all other applications at the spokes. The broker typically does the transformation and routing of messages from source application to the target application. The lack of standard message formats such as SOAP results in high maintenance costs for these interfaces as the source and target applications undergo upgrades. A slight improvement in architecture might be to use a canonical format between source and target to prevent at least one side from changing. Broker based implementations can still be moved to ESB type of functionality as long as broker can support SOAP/HTTP or SOAP/JMS transport protocols. These implementations will have to gradually evolve into an ESB by keeping the same broker at the centre of core ESB functionality but by augmenting the other

functions such as UDDI registry and Web service monitoring.

The migration path will have to be well thought out. It is not easy to change from hub and spoke architecture to SOA rapidly. Start out small by selecting one or two applications that seem easy to provide service orientation. Select a function from this application that can demonstrate service orientation and provide a service that will be easy to consume. To begin with you can start with one or two consumers however as the number of consumers increase the visibility of SOA for this particular application will drive more and more application owners to select the SOA path. For example an ERP application in a Manufacturing organization often generates Orders, Deliveries and Invoices. Typically accurate and timely order status information is required in many different applications. Exposing the status information as a service can easily attract many consumers of the service. All the consumers may not be able to invoke the service using SOAP protocols. But here the broker can add value by converting legacy formats to SOAP formats and provide service enablement for what can be called as Supported consumers. Standard consumers should be able to invoke services as it is using SOAP based transport protocols.

## Enable SOA with ESB

Once the ESB infrastructure is ready, various applications need to be encouraged to build service orientation. The service orientation of various applications will not happen overnight. A sample enterprise wide web service needs to be deployed using ESB to instil confidence among potential users within the enterprise. As mentioned in previous section select a service that will have high visibility in the organization as it might be the service that could be used by maximum applications. This deployment will also help organization to validate the process that needs to be followed to develop and deploy enterprise services. The governance issues will also come to the fore when you deploy sample enterprise service. These learning's are key to optimizing the process using which organization starts adopting Service Oriented Architecture. The architecture group within the organization needs to play pivotal role in this whole process by setting appropriate standards for naming conventions, service classifications, Security protocols to name a few. The UDDI registry plays an important role in SOA enablement. The service consumers will use UDDI to search for services available for use. The easier it is to search services in registry, the better it will make the effectiveness of the whole SOA using ESB. Hence the UDDI taxonomies need to be carefully defined for making registry searches

consumer friendly. These taxonomies will evolve as the ESB infrastructure utilization increases but it is crucial to begin with well thought of list of taxonomies.

## Conclusion

The intent of this paper is to highlight all the different angles that you need to cover when you decide to deploy a true ESB. It is directed at semi-technical users who understand the technology at high level but may not understand some of the technical details that can have impact on the overall timeline for ESB deployment. These users are directly involved with the ESB either because their business gets affected or because they may be involved in decision making process. As discussed it is quite exhaustive effort and proper organizational and technical planning can ensure that ESB is deployed in a cost-effective and timely manner. The smoother deployment of ESB should provide much needed confidence to application owners in the new technology. The right organizational environment is a key to success when implementing ESB infrastructure.

## References

### Books

David A. Chappell - *Enterprise Service Bus, First Edition June 2004*

### Web sites

Rick Robinson, (2004). Understanding Enterprise Service Bus Scenarios and Solutions Available: <http://www-128.ibm.com/developerworks/webservices/library/ws-esbscen/>

Brenda M. Mickelson (2005) Enterprise Service Bus Q&A  
[http://www.ebizq.net/hot\\_topics/esb/features/6117.html](http://www.ebizq.net/hot_topics/esb/features/6117.html)

### Copyright

Copyright © 2006 Kiran Kanetkar.

The author grant a non-exclusive licence to the Integration Consortium to publish this document in full on the World Wide Web (prime sites and mirrors) and in printed form. Any other usage is prohibited without the express permission of the author.