ISSN 1745–1884

CBDI Journal

Reprint from April 2006

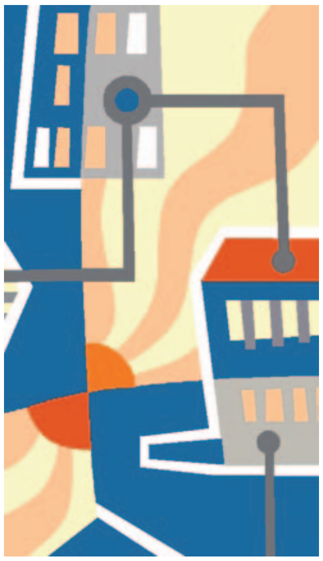
Best Practice Report

Applying ESB

The Enterprise Service Bus should be a device for ensuring policy governance. This suggests that there is more to ESB design than selecting the best product or finding the lowest cost approach reusing existing technical infrastructure components. This report looks at how the ESB can deliver varying benefits by use of layered architecture and appropriate deployment patterns.

By Lawrence Wilkes

Independent Guidance *for* Service Architecture and Engineering



Applying ESB

This report looks at the use of the Enterprise Service Bus (ESB) to support layered architecture and varying business requirements and considers the benefits an ESB can deliver. It also looks at appropriate ESB deployment patterns.

By Lawrence Wilkes

The role of SOA Middleware

SOA middleware may seem an oxymoron. SOA is meant to free organizations from the tyranny of tightly coupled implementations, which in my mind includes creating dependences on middleware, not just application and platform-level dependencies.

With support for Web Service protocols embedded in the Application Servers, and WS-STAR[1] providing support for federated, secure, reliable, transactions between endpoints, why should there be a need for additional middleware?

The reality is that existing benefits of a middleware approach still largely applies. SOA middleware can be used to:

- Separate concerns – remove middleware capability such as messaging and mediation away from applications. Though a modern platform can also assist in achieving this, such as Microsoft Windows Communications Framework (WCF)
- Support Heterogeneity – separate capability away from OS/Platform specific functions
- Provide a more agile environment where changes to infrastructure do not impact applications, or vice versa
- Form part of a shared enterprise SOA infrastructure, rather than embedded in or specific to each application solution
- Manage by policy, and manage centrally. It is easier to deploy and enforce policies through a layer of SOA infrastructure designed for this purpose.
- Web Services are not the only protocol. Even when they do become widely used, most organizations will continue to use the existing middleware and other protocols already in use for some time. Hence SOA middleware often provides support for other protocols.

A challenge for many organizations today, is that the capability required for SOA middleware and SOA infrastructure in general is that it is spread, and often duplicated across multiple products and technologies. In addition it is often found in a mixture of point solutions and specialist SOA products plus existing infrastructure that is typically upgraded to support SOA requirements. Whilst Table 1 presents some good reasons to consider new infrastructure products

[1]STAR – Secure, Transacted, Asynchronous, Reliable. Term often used in reference to the collective use of WS-RX, WS-TX, WS-Security protocols.

New	Existing
Built for SOA	Built for existing pre-SOA requirements
Based on Open Standards. Built from ground up to support open standards	Support for Open Standards. Various open standards are supported through an adaptor that extended internal proprietary platform
Built for WS-Protocols	WS-protocols are an adaptor
Support for broad range of WS-Protocols	Often only supports core SOAP, WSDL
Expose capability as Services. Using Web Services	Capability more likely exposed through proprietary API.
Native XML processing. May include XML processing optimization	XML is an adaptor
Componentized	Monolithic
Emerging, but Maturing	Mature. Optimized, high performance
Small vendor – may be acquired	Large vendor
You have to buy it	You already have the basis of it. Though requires upgrades or adaptors for SOA and WS

Table 1: Contrasting New and Existing Infrastructure Products for SOA

to support SOA, it also highlights a strong reason for many organizations to look at how they upgrade their existing infrastructure to support SOA – namely, that they already have it, and the existing infrastructure must remain in place to support existing requirements.

However, it is not as straightforward as depending on existing infrastructure. Although not an immediate concern for some organizations, SOA will place new demands on infrastructure capability that the existing infrastructure cannot so easily support. Longer term, the SOA infrastructure must itself become Service-based and able to be virtualized in the same way that is required of business capability. Even in the near term, organizations can gain advantage from using a networked approach to some SOA middleware requirements rather than using a hub and spoke approach that frequently exists today. Longer term, the federated SOA will make this a requirement.

Consequently, it is important that organizations consider the granularity of software components and the availability of Service-oriented interfaces to support virtualized, federated deployment of SOA infrastructure. This is more likely to be found in new, purpose built SOA infrastructure. This is explored later in SOA infrastructure deployment patterns.

The Enterprise Service Bus

We first considered the role of the Enterprise Service Bus (ESB) in a previous report "Time to board the Enterprise Service Bus"[2]. Since then end-user interest in ESB and vendor hype has continued to grow. I sometimes think a better expansion of the abbreviation might be "Everyone's Silver Bullet", such is the perception that all you need to buy is an ESB and all your SOA problems are solved.

In that report we identified that there was no commonly agreed definition of an ESB, or a set list of functionality one might expect to find. Consequently, so called ESB products offer a spectrum of capability as shown in Table 2.

At one extreme, ESB products may be little more than a broker, providing routing and transformation capabilities. At the other, ESB products can provide most of the SOA Infrastructure required. It is at this end of the spectrum that the most overlap occurs with other infrastructure domains. Ideally, the range of capabilities offered should be modular enabling organizations to assemble their SOA infrastructure from a range of best of breed capabilities. Unfortunately, not all vendors share a similar goal.

A well formed ESB can help organizations by providing the core mechanism to deliver SOA run-time agility. The role of ESB includes

- Providing the mediation layer between service consumers and providers to enable loose coupling
- Abstract hard coded transformation and routing of messages away from service consumers and providing resources – making the SOA easier to maintain, manage

[2]CBDI Report. Time to Board the Enterprise Service Bus? http://www.cbdiforum.com/secure/interact/2004 – 07/Enterprise_Service_Bus.php

Capability	Not quite ESB?	Lightweight ESB	Extended ESB	More than ESB?
Provision/Host Service Endpoints		✓	✓	✓
Messaging (MOM)			✓	✓
Routing	✓	✓	✓	✓
Message Transformation	✓	✓	✓	✓
Rich palette of transformation patterns		✓	✓	✓
Protocol Transformation	✓	✓	✓	✓
EAI Adaptors			✓	✓
Database lookup			✓	
Orchestration		✓	✓	✓
Security				✓
Service Management				✓
Policy Driven		✓	✓	✓
Desirable Characteristics	Implement Open Standards Componentized – enable distribution of components Extensible Service Based – Exposes capabilities as Services			
Potential Product Overlap		Orchestration Engine	Orchestration Engine MOM	Orchestration Engine MOM EAI Security Service Management System Management

Table 2: The ESB Spectrum

- Provide a central point of control for mediation and integration policy enforcement
- Host Services within the SOA itself (see Service Endpoint Hosting below)
- Depending on the ESB implementation, provide declarative mediation, dynamic mediation or mediation based on current and emerging open standards such as WS-protocols, WSBPEL, JMS, JBI
- Provide on/off ramps for multiple transport and message types
- Orchestration of Services

Service Endpoint Hosting

CBDI has promoted a layered Service Architecture as a mechanism to provide flexibility, as well as well as reuse and consistency through the use of shared Services. This is central to the Service Portfolio Planning (SPP) approach that CBDI has formalized over the last year.

A key deployment question is where each layer of the SOA should be hosted. The simple approach is to assume all Service Endpoints are hosted on the Application Server that also conveniently hosts the corresponding Service Automation Unit[3] (SAU).

[3]CCBDI Journal March 2005, SOA Reference Model http://www.cbdiforum.com/secure/interact/2004-03/SOA_Reference_Model.php

Capability	Application Server	ESB
Host Service Endpoints	✓	✓
Provide implementation of Service	✓	✓
Bind/link Service to implementation API	✓ *On the same Server*	✓
Automate mapping of native platform API used by SAU to WS-Protocols	✓	✓
Enable developer to program implementation in familiar native languages without learning XML and WS-Protocols	✓	✓
Provide a declarative approach to defining Services	✓	✓
Provide declarative specification of rules, processes, mediation		✓
Broker Service requests to alternate Endpoints		✓
Bind/link Service to SAU hosted elsewhere		✓
Transform Service Requests to different Implementation Format		✓
Provide dynamic, content-based mediation		✓
Policy Driven	*Config file*	✓

Table 3: Application Server and ESB Comparison

However, it is not necessary or even desirable to have such a close physical, tightly coupled relationship between the Service Endpoint and its SAU. The Service Endpoint should be a first order concept that is independent of the SAU, ideally with the two only linked dynamically at runtime based on prevailing policies.

Most modern Application Servers can also act as a Service Broker, with a configuration file maintaining the link between the Service Endpoint and the SAU. However there are limitations to this, namely that the configuration in the Application Server is unlikely to recognize and link to SAU's that are hosted elsewhere, or broker Service Requests to other Service Endpoints that are also hosted elsewhere.

Table 3 provides a comparison of typical Application Server and ESB in terms of their ability to host Service Endpoints and act as a Service Broker. With this in mind, Table 4 considers the role of the Application Server or ESB to host each of the Service layers.

Figure 1 visualizes the possible roles of the ESB and the Application Server in hosting Services. It shows

- The ESB hosts the Process Service Endpoint. The SAU for this is implemented using the process orchestration capabilities of the ESB (typically WSBPEL)
- The ESB also hosts the Business Service Endpoint.
- The Application Server hosts the SAU for the Business Service Endpoint
- The Application Server hosts the Underlying Service that provides access to the SAU (via it's API)
- The ESB provides the transformation and mediation capability required to mediate between the Business Service and the Underlying Service

Delivering SOA Benefit Patterns

As well as being suitable for hosting Service endpoints for certain layers, the ESB is also particularly suitable for delivering certain SOA benefits patterns that we looked in the recent CBDI Journal report[4]. The advantage offered by using an ESB to implement these patterns is shown in Table 5.

[4]CBDI Journal March 2006, SOA Benefit Patterns: http://www.cbdiforum.com/secure/interact/2006 – 03/SOA_Service_Benefit_Patterns.php

Service Layer	Application Server	ESB
Process Services	Host Process Service Host SAU as Process Component to implement logic, rules Process implemented as code.	Host Process Service Host SAU as process logic, rules scripted (declaratively) in ESB. Use Orchestration capability to implement process
Business Services	Host Business Services Host SAU as Business Component to implement logic, information access	Host Business Service Host partial SAU – validation, some business rules. Business rule driven mediation Mediate messages to SAU in Application Server
Underlying Services	Host Underlying Service Host SAU in existing applications or packages	Mediate to Underlying Service Provide transformation capability. Adaptors to existing systems and packaged applications
Utility Services	Host Utility Service Host SAU as Utility Component	Mediate to Utility Service Host some utilities where provided as capability of ESB – e.g. Data Transformation Utility

Table 4: Hosting Service Layers

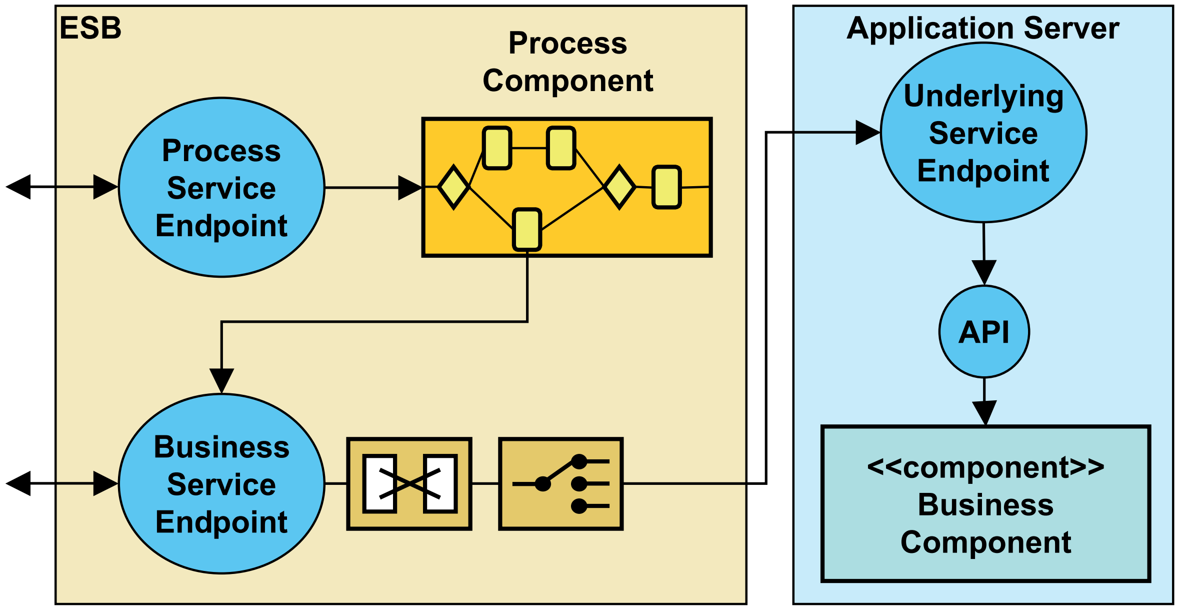

ESB
Process Component
Process Service Endpoint
Business Service Endpoint
Application Server
Underlying Service Endpoint
API
<<component>> Business Component

Figure 1: Role of ESB in Hosting Service Endpoints

Pattern/Strategy	ESB Advantage	Use of ESB
Façade	✓✓✓✓	Extended ESB have adaptors to existing systems and packaged applications, with extensive transformation capability
Single Service	✓✓	The real ESB advantage is where the Single Service is also a façade, which is often the case
Standardized Service	✓✓	An ESB implemented on a truly enterprise basis can ensure compliance with standardized services across the enterprise
Standardized Semantics	✓	An ESB implemented on a truly enterprise basis can help ensure compliance with standardized semantics across the enterprise
Commodity Service	*None*	*The ESB offers no specific advantage in delivering this pattern*
Common Component Service	*None*	*The ESB offers no specific advantage in delivering this pattern*
Multi-Channel Service	✓✓	ESB can offer an advantage where the multi-channel Service is a façade across existing channel specific Services
Real Time Service Behavior	✓✓	The ESB cannot itself transform existing systems to provide real-time behavior but could play a useful role in supporting real-time interaction between Service Provide and Consumer, and providing a real-time façade over the existing systems. Support in the ESB for WS-STAR would be desirable where required for some real-time messaging patterns.
Real Time Mediation	✓✓✓✓	ESB's excel of course in providing real-time mediation, with support for policy driven mediation, and declarative approaches make it straightforward to configure
Differentiated Service Behavior	✓✓✓	The ESB can again provide policy-driven mediation to provide differentiated services behavior, and route requests to different service automation units where applicable

Table 5: SOA Benefit Patterns Supported by ESB

The goal should not be to have a single physical ESB to ensure consistency, but a single logical ESB where policies and rules can be distributed and executed locally, but administered centrally.

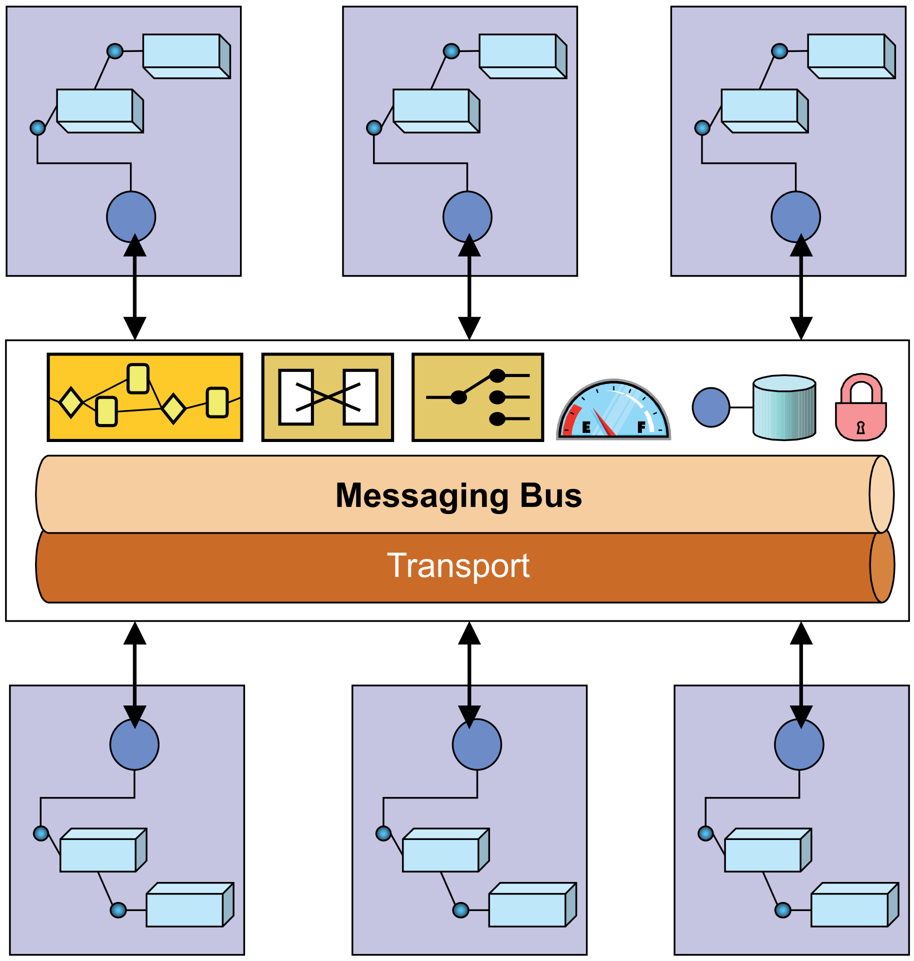


Figure 2: "Classic" ESB Diagram

ESB and SOA Infrastructure Deployment Patterns

The ESB is often depicted as in Figure 2. The bus mediates messages between the various Services that "plug" into the bus.

However, the notion of a "bus" is perhaps misleading when considering the physical deployment of the ESB. Rather, there are various patterns of deployment that are typically used.

Hub and Spoke

Often the ESB is deployed in a hub and spoke style that is reminiscent of Enterprise Application Integration (EAI) tools. This is not surprising given the heritage of some ESB products.

Here the core ESB and other components of the SOA infrastructure are deployed centrally and all the messages are routed to it for processing. This is a sensible approach where the Services and Resources are themselves centrally located. It is also useful where organizations are seeking to connect large numbers of existing non-SOA systems where the ESB is seen as an evolution of their existing EAI approach, and hence implemented by extending their existing EAI infrastructure.

Network

The opposite extreme is the network pattern, where each node contains full range of ESB and SOA infrastructure capability.

This is appropriate for an SOA that involves distributed, federated scenarios such as in a large global organization, joined up government across multiple departments and agencies, or ecosystems of collaborating business partners.

The key benefit of this is that Service Requests can flow directly between participants in a federated scenario without having to pass through a central hub which becomes a potential bottleneck and single point of failure. Mediation can be implemented locally which distributes processing, or can also be undertaken step by step as a Service Request passes through multiple nodes on its journey from Service Consumer to ultimate Service Provider, with each node applying appropriate polices.

The downside is that every node needs a full set of SOA infrastructure capability, which may be costly in terms of both resources and product licenses, and configurations and policies may require synchronization.

The network approach may appear to share many of the problems inherent in point-to-point integrations. However the ESB specifically overcomes these issues:

1. by implementing a standards based proxy layer – creating loosely coupled connections
2. by enabling central definition and local deployment of policy to ensure consistent response to events.
3. **or by enabling sharing of policies and configurations between relevant participants (if there is no central authority)**

Given the federated participation of some SOAs, there may be little alternative to the network approach as there is no shared infrastructure or obvious hub, or no desire by the participants to nominate any one participant as the hub.

Combined

Many organizations may find that the ideal pattern is to combine both the Hub & Spoke and the Network pattern, for example with lightweight SOA Infrastructure nodes, together with a heavyweight centralized core of capability. For example, initial mediation processing may take locally to format the Service Request and address it to the appropriate endpoint, but it is still sent to the hub where additional transformation takes place and the request is re-routed to an alternative endpoint (based on policies that were not apparent to the dispatching node). Or policies may dictate that some Service Requests can be routed directly to another node on the network, whereas others must to dispatched to the hub for processing.

Business Service Network

A variation on these patterns is the Business Service Network[5] where the SOA Infrastructure hub is provided by a 3rd party provider. This could be used as a purely hub & spoke approach where all messages are sent to the hub for processing, or with the combined approach where some messages processed locally and sent directly to other nodes.

This may be appropriate for example to centrally managed ecosystems where participants are happy to delegate SOA infrastructure responsibility to an independent provider.

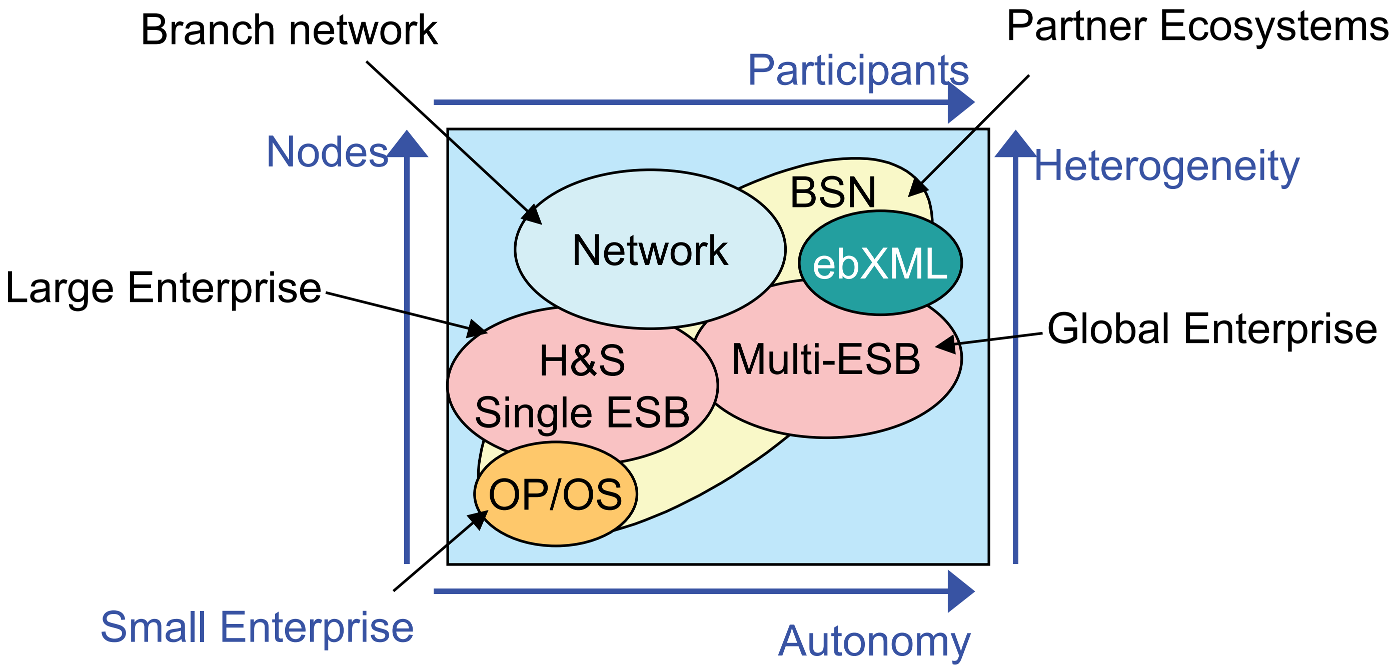


Figure 3: ESB Deployment Profiles

[5]http://roadmap.cbdiforum.com/reports/ICT/

Figure 3 profiles these deployment options by considering factors such as

- The number of Nodes that might require local capability
- The number of different participants
- The autonomy of participants
- The diversity of the platforms in use (heterogeneity)

It also introduces other options for providing ESB and SOA infrastructure capability such as

- **OP/OS –** the operating platform or operating system provides SOA infrastructure and ESB capabilities. Small enterprises may find that the Windows Server platform provides all the capability they need internally. Then to complement this, where they require interaction with external participants they may decide to delegate this to a BSN provider
- **ebXML –** in some B2B scenarios the participants might agree to standardize on infrastructure that complies with the ebXML standard. However, this would normally only apply to external B2B activity, not the internal SOA infrastructure
- **Single vs multi-ESB deployments**. It might seem oxymoronic to talk about an enterprise having several ESBs. However, it possible that in large global enterprises where there is considerably autonomy between divisions, it may be effective to deploy multiple ESB's (a Divisional Service Bus?) and to treat other divisions as effectively external participants each with their own ESB, rather than to try to force the entire enterprise to share a single infrastructure. The challenge would be in ensuring compliance with truly enterprise-wide policies if there was a diversity of ESB products in place.

Summary

The ESB can play a useful role not just in hosting Service Endpoints but ensuring policies are enforced. An ESB implemented on a truly enterprise basis can provide a single point of consistency and policy execution for all Services, both those provided and consumed by the organization.

However, organizations must be careful to ensure the ESB is not a bottleneck or a barrier to delivering agile SOA. Hence ESB deployment patterns are important. The goal should not be to have a single physical ESB to ensure consistency, but a single logical ESB where policies and rules can be distributed and executed locally, but administered centrally.

Independent Guidance *for* Service Architecture and Engineering

Subscribe to the CBDI Forum

The CBDI Journal *is published monthly with a combined July/August edition. An annual corporate subscription includes access to all back numbers plus access to Powerpoint Libraries and the CBDI Hot Line service.*

For more details see: www.cbdiforum.com

CBDI Objectives

CBDI Forum aims to provide independent, action oriented practice guidance on Service Oriented Architecture and Component Based Development for architects, business analysts, project managers, designers and others involved in creating and delivering advanced architectures.

CBDI Delivery Channels

CBDI Forum provides:

- Subscription services – continuous practice guidance published in the CBDI Journal every month (with July/August combined into one volume)
- Workshops and Seminars – providing indepth education on architecture, process and practice. Public and In-house classes are available.
- Consulting – specific guidance on adoption roadmap including status assessments, methodology customization, architectural guidance including reference architecture development, governance reviews, business design and strategy development.

CBDI Background

CBDI Forum is the Everware-CBDI research capability and portal providing independent guidance on best practice in service oriented architecture and related delivery processes. Working with F1000 enterprises and governments the CBDI Forum research team is progressively developing structured methodology and reference architectures for all aspects of service oriented architecture.

A CBDI Forum Subscription provides a corporation or government department with access to a unique knowledgebase, ongoing continuous practice research guidance materials and hotline access to CBDI Forum experts. The monthly CBDI Journal provides in-depth treatment of key practice issues and guidance for architects, business analysts and managers. Forum Meetings are held periodically in Europe and North America allowing peers to engage and exchange experience and best practices.

CBDI Forum is the Everware-CBDI practice research capability and portal

Contact Us

For further information on any of our services contact us at: info@cbdiforum.com or +353 28 38073 (International)