TORONIX

Ahead of the Pack – The Source for High Performance WebSphere SOA

Editor: Robert R. Rowntree, M. Eng., MBA, PMP, CISSP, P.Eng. *Summer 2007*

In this Issue

WebSphere SOA Performance Engineering

Tracking down tough performance issues across complex systems and applications is likely not an easy task. For new applications, it seems projects are always 98% complete for a long period of time. Often, the last 2% of the project includes the toughest part of the project that requires a specialized skill set - performance engineering. Click here for more info

High Performance SOA Part III – The Global Enterprise ESB

Adding an Enterprise Service Bus (a.k.a., ESB) solves some very ugly problems for Enterprises - the "rats" nest of application-to-application connections that **explodes maintenance costs** while destroying the organization's **ability to change fast**. If you are in a slow industry that does not have to change much, it may be a mute point. Also, ESB centralizes the ability to translate messages between two parties much in the way a Chinese man needs to have his language translated for an Italian speaking woman – all on the "fly" and in real-time. This allows partners both internal and external to not have to change the way they interface to other systems – just create the translators in the ESB and your feeds from partners are ready to go – very slick, very fast. And when you have to add/drop/change partners, an ESB makes this easier because the changes are centralized. Quality of service is easier to manage as well. If you want multiple service providers, one is fast and costly and one is not so fast and reliable but cheap, you can embed the level of service that is required right in the message. For example, if income is over $200 thousand use the faster processing service provider. The ESB will route the request in real-time to the premium faster service provider.

To cope with a global enterprise, multiple ESB may have to be incorporated in your organization depending on the degree of business decentralization, geographic dispersion and/or LOB structuring. You would do this in order to minimize network bandwidth requirements and maximize speed of information delivery. There are two choices for an ESB either WESB or WMB (a.k.a., WebSphere Message Broker) or in some cases both may be required. The WESB excels where Web Services requirements predominate while the WMB dominates where complex legacy interaction is required – in many global enterprises, both would likely be incorporated into an ESB solution.

The line up for these series is as follows:

- **Part I – Business - The Business of SOA - (Winter 2006)**
- **Part II – Technical – Built-in WAS SOA – Web Services (Spring 2007)**
- **Part III – Technical – ESB Advantages and Options**
  - WESB - WebSphere Enterprise Service Bus
  - WMB - WebSphere Message Broker
- **Part IV – Technical – Business Process Engineering** - WebSphere Process

Contrarian – Communist IT – 2 Users costing $3 million/yr

"We need a complete review of this whole StoreManager portal solution. We need a portal expert to look this over" barked the Director of IT at Mr. BigGrocer (fictitious name) in the US. "We just spent another 3 months and $100 thousand dollars on fixes to that application and the performance still sucks". When I got there, I noticed a few peculiar things. One, the application group said that at any given time there were 2500 store managers accessing the system. "That was pretty good" according to the developers but it was still slow. How slow? Well, those 2500 users were getting 10-20 second response time - way too long. However, when I turned on the TPV analyzer in fact, there were only 2 users at any time actively using the system. "Can't be, no way, that TPV is wrong," said the angry and hostile developers and their managers. To confirm my results, we get the Mr. BigGrocer test group to analyze the system for active users with Wiley-CA Interscope. Sure enough, there were only 2 active users at any given time on the multi-million dollar system with about 2500 users logged in. $3 million dollars worth of hardware, software and support staff. Note, the CEO had mandated that store managers must login to the StoreManager portal- "you can lead a employee to water but you can't make him think" syndrome. It also included various backend datamarts, online databases and a terabyte data warehouse – the data warehouse system is on forty – 4-way machine complex.

The root technical cause was a small database on a 2-way system that was taking 97% of the total response time. There were no priorities set on who could access that system and it became overloaded. Jerry in customer service could run a dumb query on "number of disabled employees who eat Twinkies at midnight" and knock out all the store managers wanting to know critical payroll information. Everything was free (communist IT) so the system got overloaded. The $100 thousand re-design effort of the StoreManager Portal database access components was not the right solution and was discarded. The identification of the problem was masked by the development group in their apprehension in confronting the database group and tried to work around them. The company really needed to set priorities at the business level and then implement them so resources utilization matched those priorities – that is, Jerry should get limited access while the store managers receives the highest priority. The LOB responsible for the overloaded database took the simple approach; they increased the hardware significantly, as well as, looking into prioritizing access so that in the future the system was not swamped with low priority activities.

SOA – Recap – In Your Business, What's Slowing You Down?

Research overwhelmingly has demonstrated that the fastest moving markets where a small edge can mean dramatic revenue gains include the finance, capital, distribution and insurance industries. What slows you down is processes that are not well understood can neither work nor change fast. Compounding this is **expensive activities that are "baked in" with low value activities** that impede the re-allocation to either lower cost IT, lower cost labor or both. Think outsourcing low value activity, but only if you can break the negative "baked in" approach. Currently, one of the best ways to "un-bake" your enterprise for high performance is through SOA. SOA process re-engineering success factors include tech savvy management, engineering versus just coding SOA, increased centralized SOA management

and enterprise versus LOB based profitability incentives. The key question to ask is: Will LOB's pay for the services or will access be free? Free access will likely result in over use of your services and associated resources. In addition, there is added cost for accounting related to charging for services. **Bottom line is either *you move faster* or *your customers will move* to faster competitors.**

SOA – A Quick Technical Review

SOA involves the orchestration of repeatable business tasks known as services. The top technical drivers for SOA are **standards**, the pervasive ability to **discover services** and widespread **interoperability** between systems and applications. Although SOA is technology neutral, the best technology currently available to implement it is Web Services coupled with an ESB such as WebSphere Message Broker V6 and/or WebSphere ESB V6. The core of Web Services consists of a requester, provider and a service broker as shown in Figure 1**.** The service broker allows the discovery of services by the requester. The provider publishes its services so requesters can find them. In the case of a *static* Web Services, no service broker is involved and the addresses are "hard wired" into the requestors. This causes less desirable by albeit simpler coupling characteristics. Alternately, *dynamic* Web Services provides service discovery through the service broker.

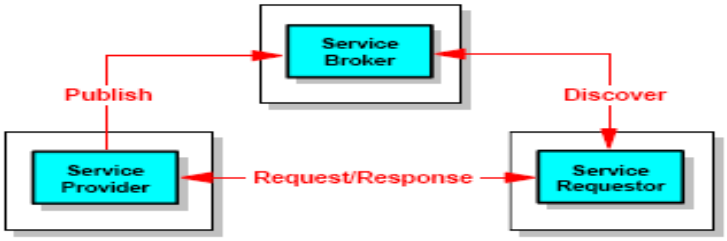


Figure 1 Service Oriented Architecture fundamentals

In the context of WAS V6.1, the functions of the Service Broker can be accomplished by the built-in UDDI registry. Although not shown on this simple diagram, the built-in SIB could be used within WAS to mediate and transform requests between providers and requestors. For example, transformations could include changing a provider's XML formatted message into a format the requestor understands. Lastly, Figure 2 shows the IBM products and their positioning within the reference architecture.

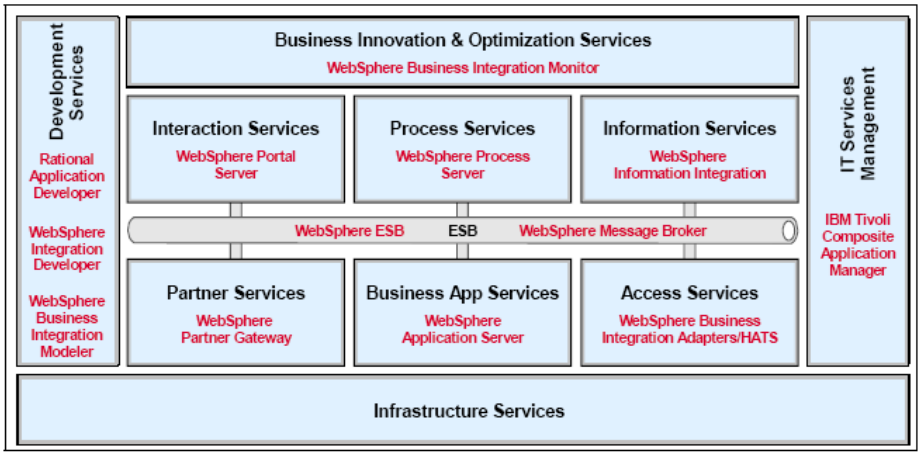
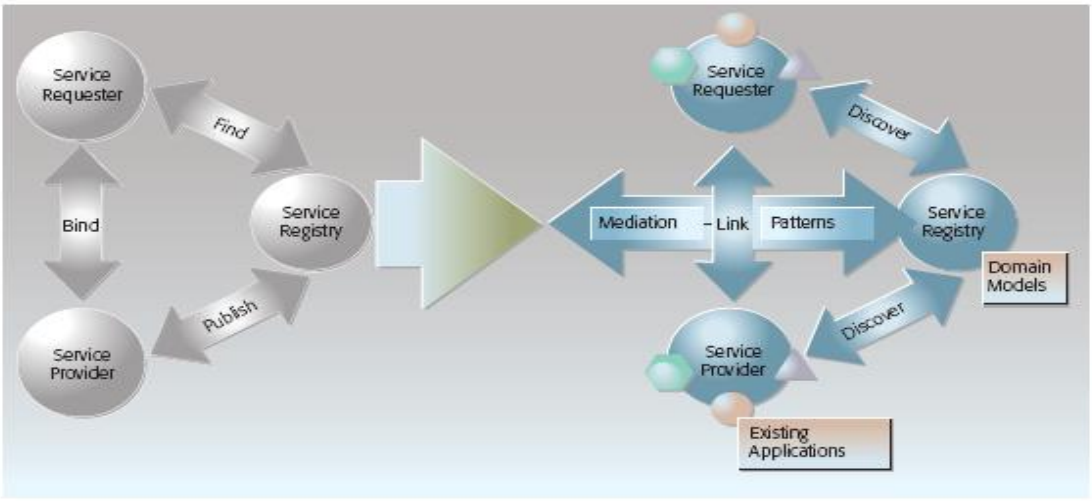


Figure 2 SOA Reference Architecture with Product Mapping

ESB 101 - What is an Enterprise Service Bus?

An ESB provides the connectivity layer between services requestor and service provider as shown in Figure 3. Unlike the Web Services requirement on the left, the inclusion of an ESB will allow almost any application via any protocol to make a distributed request to a provider without knowing its "true" destination. For example, one application makes a JMS request to the ESB and the ESB then makes an EIS request to a mainframe – the ESB does the protocol conversion seamlessly. Although, 50 applications may make request to this mainframe provider, if the location changes it needs to only update addresses in one spot, the ESB. This increases the maintainability of the application while lowering the lifecycle costs. In addition, message content can be modified (mediation).



Web Services + Custom Binding Web Services + ESB (Mediation)

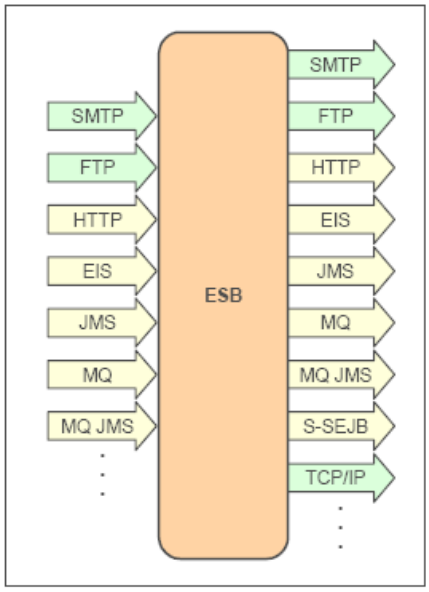
Figure 3 Service Oriented Architecture With/Without an ESB

Patterns - ESB Usage Patterns

Usage **patterns** as shown in Figure 4 and Figure 5 represent broad usages of the ESB and include:

1. **Service Routing** – the simple case where a request is routed to one end target. In some cases, routing is done to send portions of a message that are relevant to the end provider. For example, a travel package is split into 2 requests – one, a car rental requests to a car vendor and a hotel requests to hotel vendors.

2. **Switch Protocols** – requestor and provider use different network protocols. For example, switching the protocol from HTTP to TCP/IP. Dynamic routing facilities in the ESB allows service providers to be enabled and disabled so that new providers can be dynamically added or deleted as business partnerships agreements change.



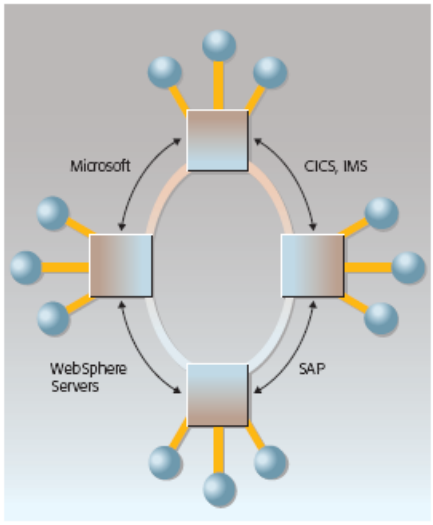
Switching Protocols

3. **Publish and Subscribe** – one request is distributed to multiple target provider. For example, the company has multiple (say, 20) hotel suppliers and a quote is needed for the best value/price for a given set of room requirements.

4. **Message transformation** – Changing COBOL copybook data to XML or an XML format to a provider's XML format.

5. **Match Making** – Requests are routed to provider services based on dynamically changing policies set in the ESB. For example, several of the same services are attached to the bus but the highest capacity and preferred provider is not available. The request is routed to a lower throughput provider.

Figure 6 shows a practical example where service requestors have their messages transformed and routed by and ESB before delivery to the service providers – Application Servers and WebSphere MQ.



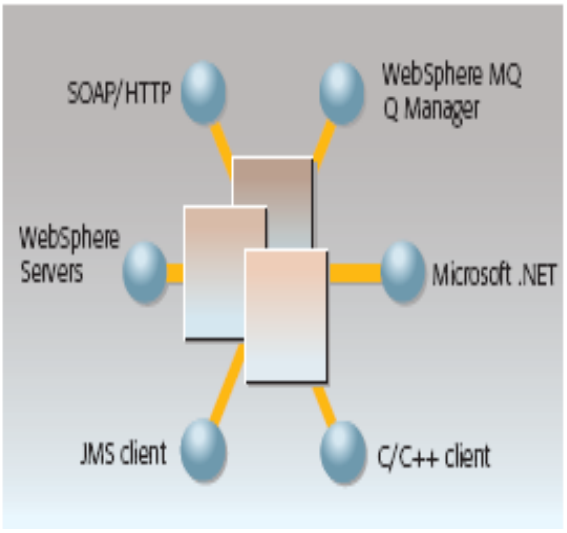
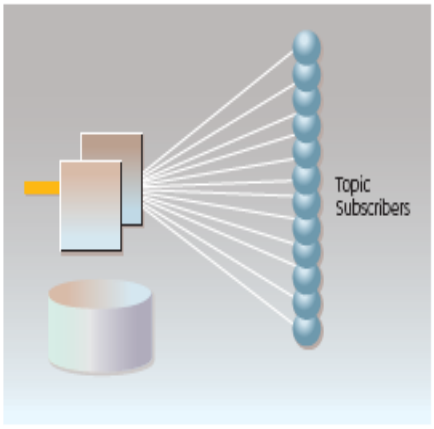


Figure 4 **Service Routing** **Protocol Switching**



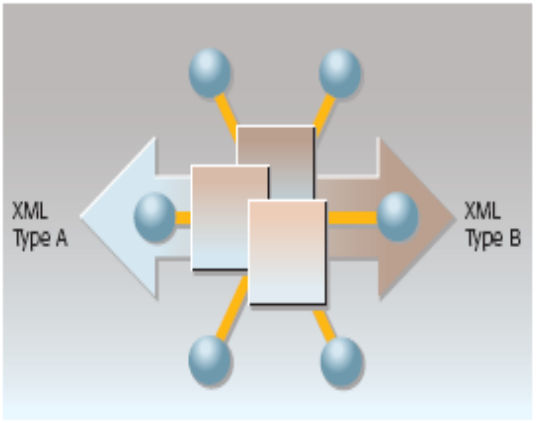


Figure 5 **Publish & Subscribe** **Transformation**

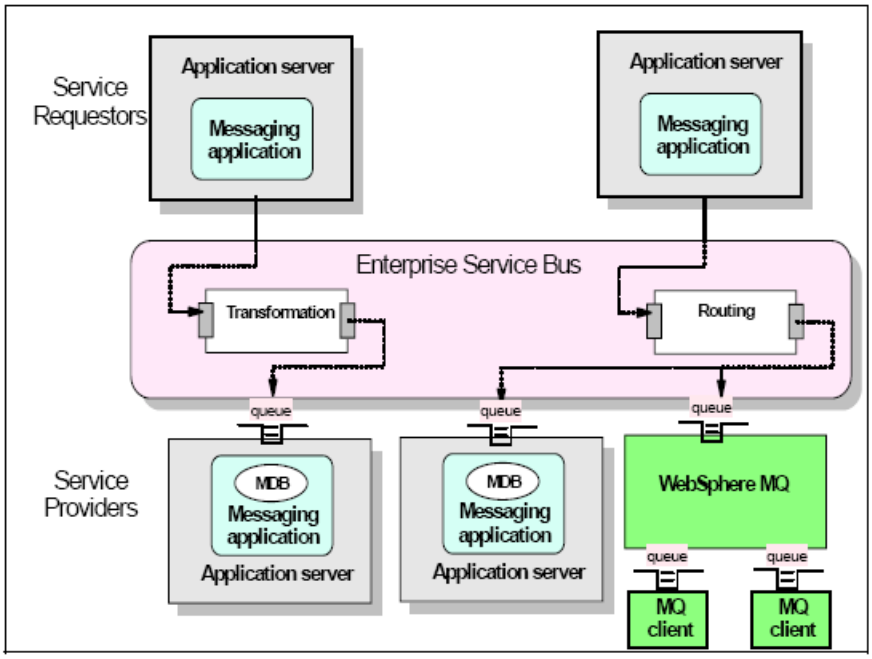


Figure 6 WebSphere example showing ESB transformation and routing

Mediation - ESB Mediation Patterns

Mediation as shown in Figure 3 is the process of modifying the message (payload) or associated message header (context) information to:

1. **Monitor** – Monitor services levels or assist in problem determination.
2. **Transcode** – Modify the payload from one format to another. SOAP to Financial SWIFT format.
3. **Modify** – Add, delete and/or modifying data elements for provider delivery. This could include encryption and adding results of a database query while the message is on-route.
4. **Validation** – validate the format and content. Decide whether it should be delivered to the destination.
5. **Router** – Modify the route of the request based on data in the message. For example, customer service requests that are critical get routed to one service, less critical requests get routed to a lower priority service end point.
6. **Discovery** – Request the latest services available from the ESB repository of current services.
7. **Aggregator** – A request is made for a hotel, car and airlines for a packaged trip. The aggregator intelligently waits for all the quotes to arrive back from the providers, consolidates the message and returns it to the requestor.

ESB 201 – ESB Strategy and Architecture

The corporate organizational architecture will typically drive the technical architecture that incorporates an ESB. In the simplest form there is one enterprise wide ESB as shown in
Figure 7. These configurations would typically be for a highly centralized organization or a mid-size organization with likely one geographical location or a pilot based ESB implication typically for one LOB. This configuration exposes the ESB for communication with external partners.

The followings are SOA components:

- **SOA Firewall** – Acts to strip out risks due to malicious or accidental mal-formed message payloads typically in XML. Web Services typically tunnel through http in corporate firewalls. The IBM Datapower SOA firewall provides XML/SOAP based filtering on headers, body and network layer data. Also, Datapower products includes schema validation, message form validation, verify digital signature and more.
- **ESB Gateway** – The gateway component should act as a proxy for the ESB zone. It provides controlled access in terms of security, auditing and forwarding of requests to the right hub – if there are multiple hubs. Practically, with smaller implementations this capacity could be served by the product constituting the hub. This can be software based products such as WMB, WESB or high performance systems such as WebSphere Datapower products that specialize in XML processing.
- **Partner Gateway** – This is another gateway for interactions that are based on higher level, business standards versus lower level, technology based standards that an ESB handles. The partner gateway handles EDIINT AS1/2/3, chemical industry CIDX and medical systems ebMS to name a few. Typical ESB standards handled are SOAP, HTTP, JMS and EIS. WebSphere Partner Gateway offers these capacities.
- **Hub** – This component has the core integration capacity that performs the protocol switching, mediation and transformations. Depending on requirements, the hub could be served by the WMB and/or the WESB.

Sometimes the requirements for the ESB cannot be handled by one ESB product. For example, many large organizations may need advanced Web Services with 2 phase commit capacity as well as legacy access capacity. In this case, both WESB and WMB would be implemented – WESB would handle the 2PC Web Services and WMB would handle the legacy access.

Single ESB - WMB as the ESB with External Exposure

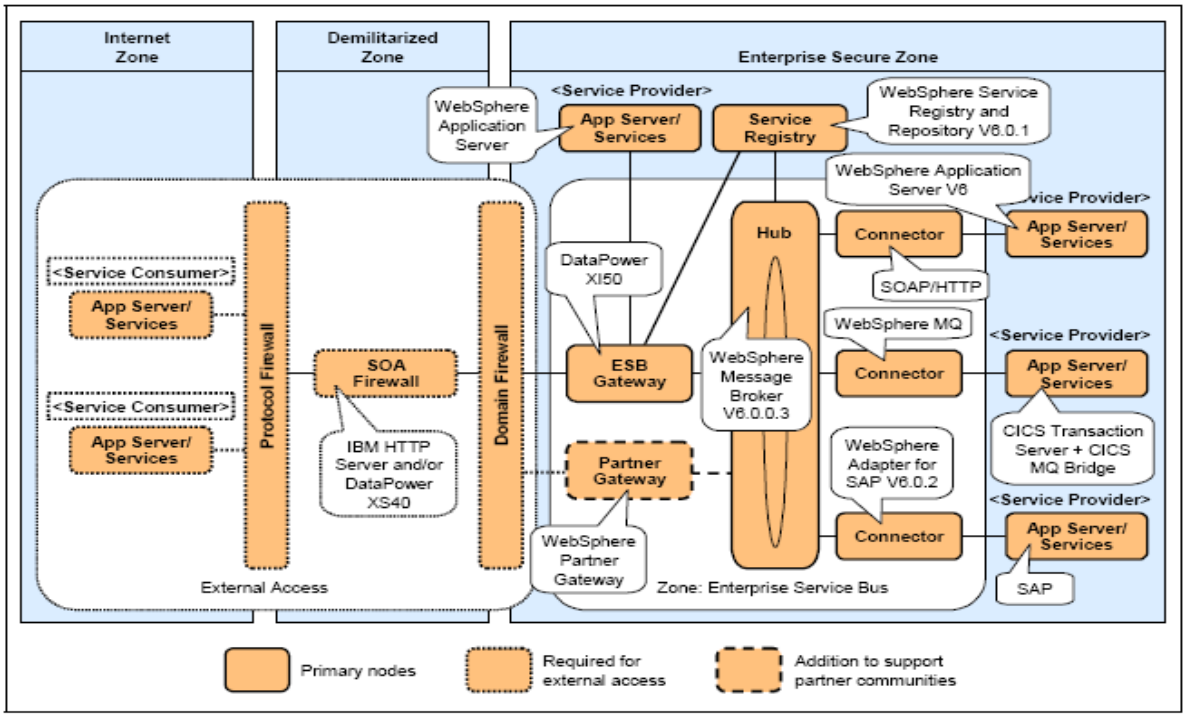


Figure 7 Basic ESB Scenario – Single ESB with WMB as the ESB with External Exposure

ESB Consideration for Large Global or Complex Organizations

For most **large organizations**, various dimensions of the organizational architecture affect the ESB architecture including business strategy, geography and technology selection. If **business strategy** calls for a highly decentralized organizational structure perhaps to serve market nuances better, at least one ESB per LOB would likely be the key to the architecture as shown in Figure 9. Messaging relevant to a LOB would stay within the unit. This would limit the interactions between business units to those interactions where multiple LOB's need to be involve such as a large cross LOB deal. Also, one unit may be in a highly profitable business and demands flexibility while another may be in a commodity business where cost is critical. In the latter case, lowering integration costs would be the driver and the former would be for speed of customer value delivery. Operating across **wide geographic areas** may impact system's throughput due to slow local international networks. In this case, ESB functionality is handled locally versus all messaging forwarded to a centralized ESB at head-office. Each major business unit may have some latitude in technology selection. Therefore, each unit, ESB may be from a different vendor and have to integrate.

There are four ESB Architectures each used to in conjunction with a business's overall strategy and associated structure:

1. A global ESB
2. Directly connected ESB

3. Brokered ESB
4. Federated ESB

These options are shown in
Figure 8 and the pros and cons are laid out in Table 1. Figure 9 show the direct connect approach, Figure 10 the brokered configuration and
Figure 11 a brokered approach.

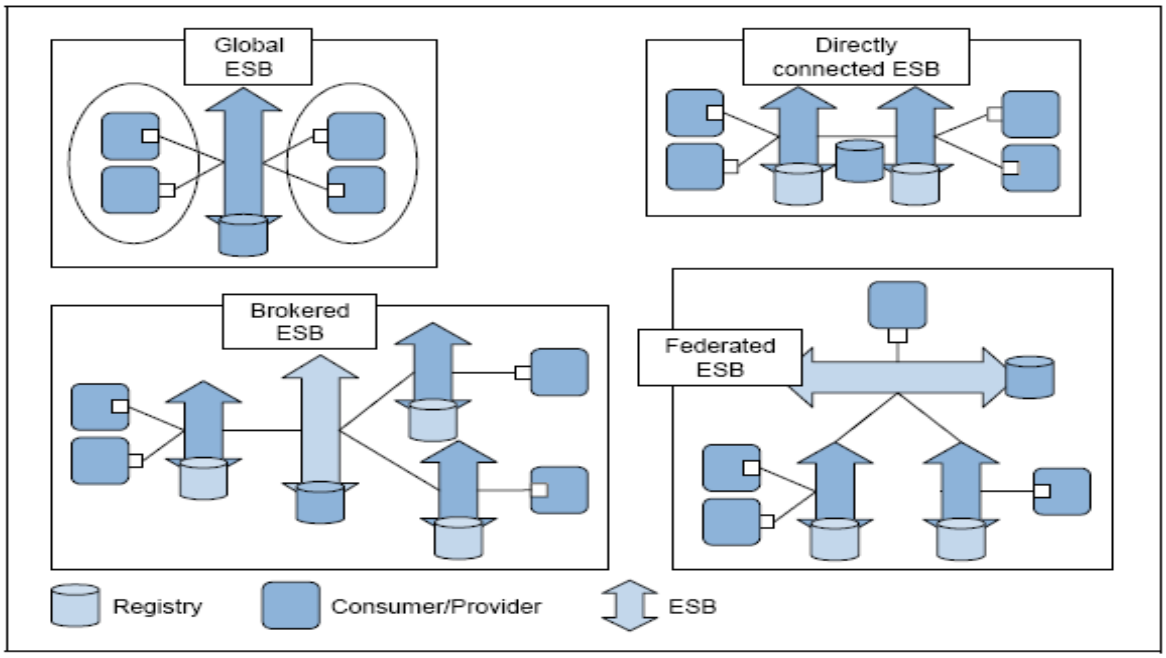


Figure 8 Options for Large Enterprise ESB Architectures

Best ESB Architecture for Different Business Strategy				
Option	**What**	**Best Use**	**Pros**	**Cons**
Global	Single ESB All services visible to all requesters	Department or Small Enterprise where all services are used throughout the enterprise.	1 centralized administration	Does not scale well across geographies.
Directly Connected	Services provided and managed by each LOB. Services available Enterprise wide	Enterprise that is largely decentralized in structure. Distinct LOB with own accountability with only limited requirements to	Complete Decoupling - Neither domain is aware of connection.	As LOB's and their ESB are added, ESB point-to-point becomes a problem. Copies of Domain 2 services that need to be exposed in Domain 1, need to be

Best ESB Architecture for Different Business Strategy				
Option	**What**	**Best Use**	**Pros**	**Cons**
		follow or use centralized resources and standards.		copied to Domain 1. Maintaining multiple ESB interfaces brings us back to the same problem we were trying to solve. Albeit, we will have hubs of points-to-point instead of single connections of point-to-points. Monitoring, auditing and problem determination are largely decentralized
Brokered	Master ESB acts as gateway into an organization for SOA. Selective exposure of services between LOB and their ESB's	Need for decentralization but Enterprise also requires sharing of services across LOB. Master ESB provides routing, security and monitor but no access to back end systems directly.	Service registry can consolidate view of all services. Need to only change service endpoints in one place.	Central registry must have replicas copied to other participating domains. Overhead in having to go through multiple gateways and just 1 hub.
Federated	Extension to Brokered ESB but consumers can connect directly with the central hub. 1 Master ESB with several slave ESB	Enterprise wide business processes need to access services offered by LOB's ESB's. A composite of services is needed so federated access is best way to provide the composition. Used by organizations that want to federate a set of moderately autonomous departments under the umbrella of a supervising department.	Highly scalable No double hops through gateways and hubs.	Complex requirements on gateway and hub nodes.

Table 1 Comparison of 4 Possible ESB Architectures vs. Business Strategy and Organizational Structure

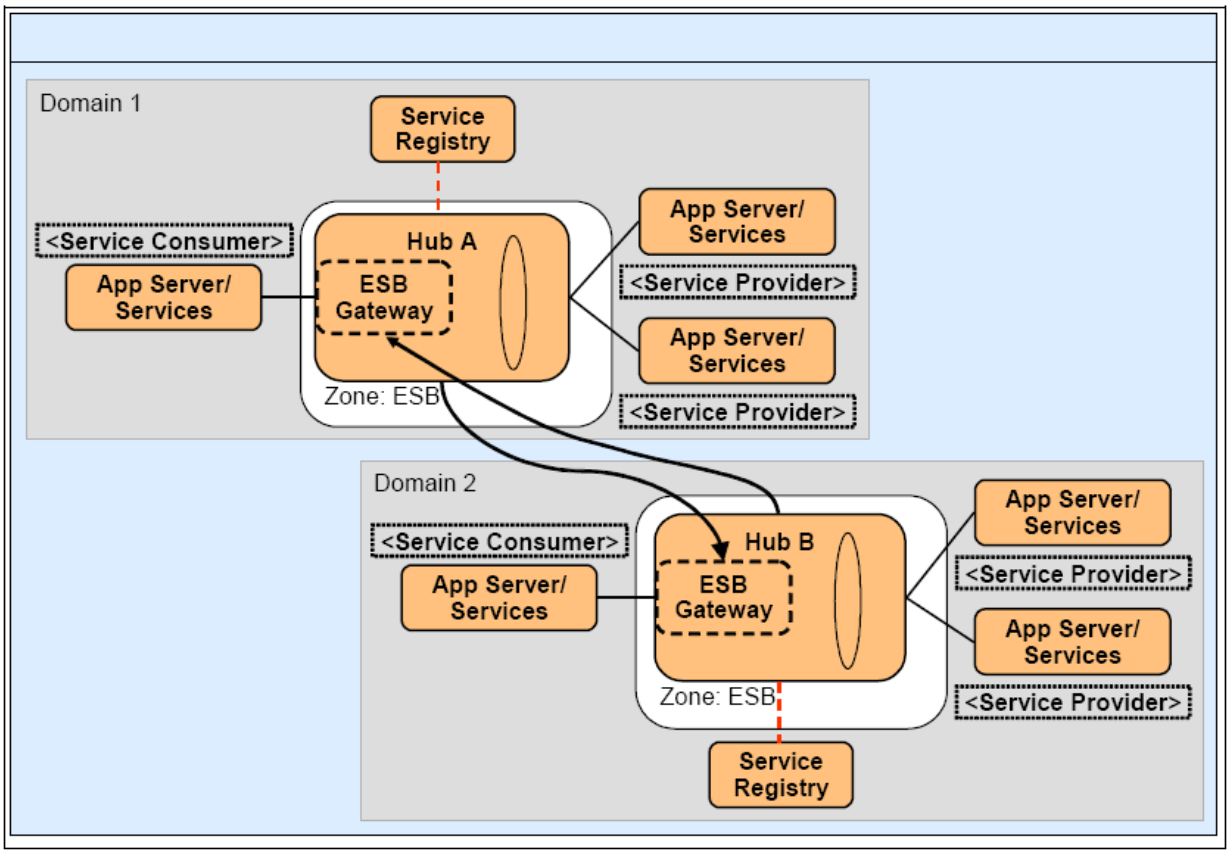


Figure 9 Direct Connect

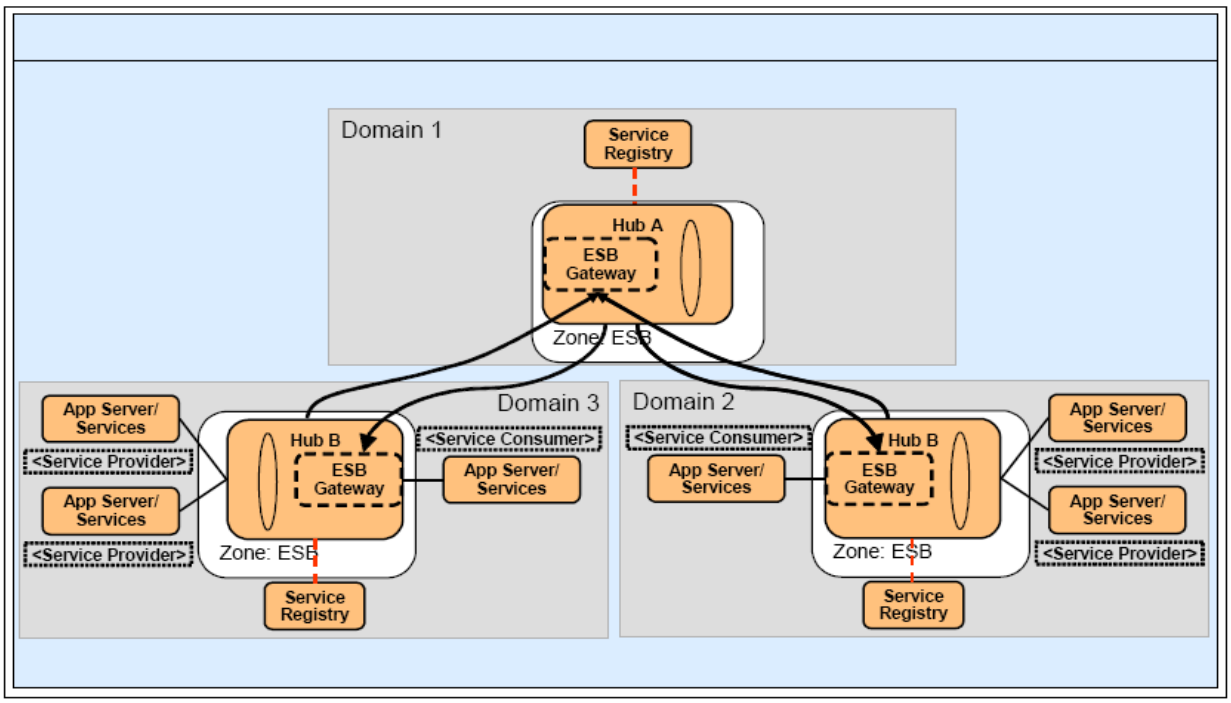


Figure 10 Brokered ESB

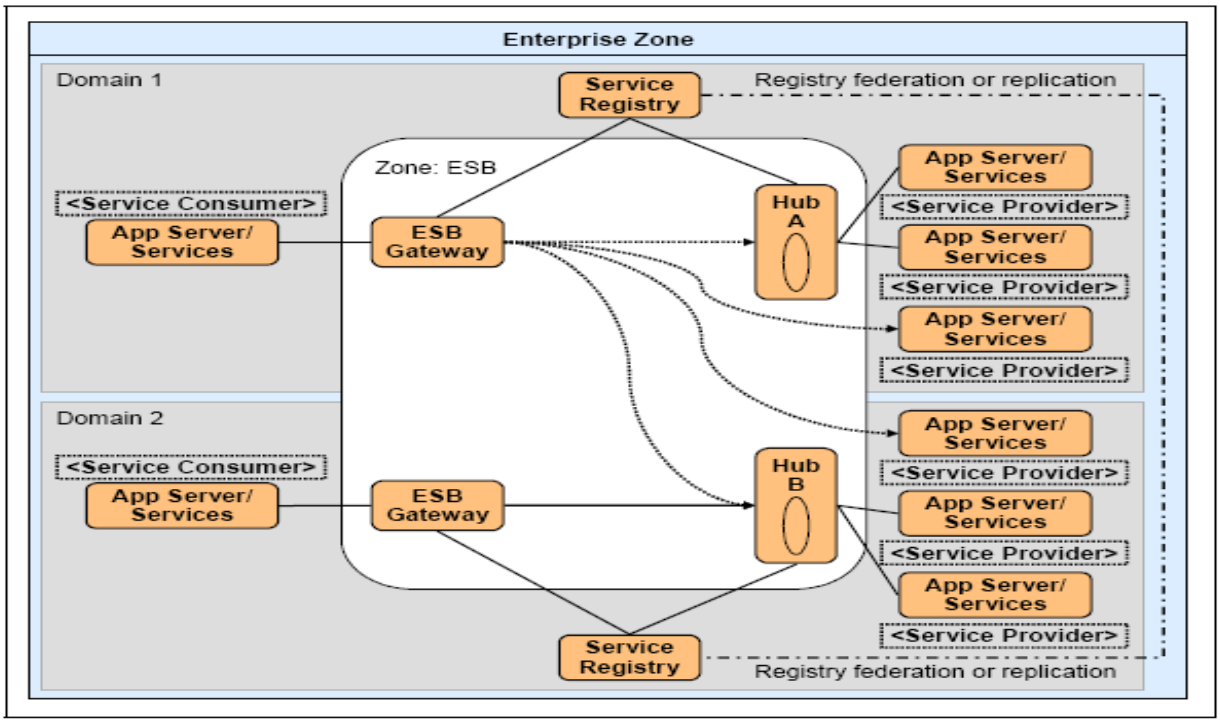


Figure 11 Federated ESB

ESB 301 – Two IBM ESB Choices – WESB or WMB Both

The WESB is primarily directed for use in pure Web Services environments while the WMB serves Web Services and many other enterprise protocols. Figure 12 shows an overview of all the major components and interactions of the WESB runtime. WESB applications are built with the WID (WebSphere Integration Developer). While it depends on the SIB in WAS it can also involve messaging to WMQ through either the MQ or JMS API. Clients can be C++, .Net or Java. There are many WebSphere Adapters including popular ones for SWIFT, FIX for financial markets and SAP and PeopleSoft adapters for enterprise-packaged software (see detailed chart below in Table 4).

WMB is shown in Figure 13. It shows a product that includes Web Services, connectivity to real time devices, telemetry sensors and mobile capacity. It also has the capacity to reach out to enterprise resources such as CICS and VSAM.

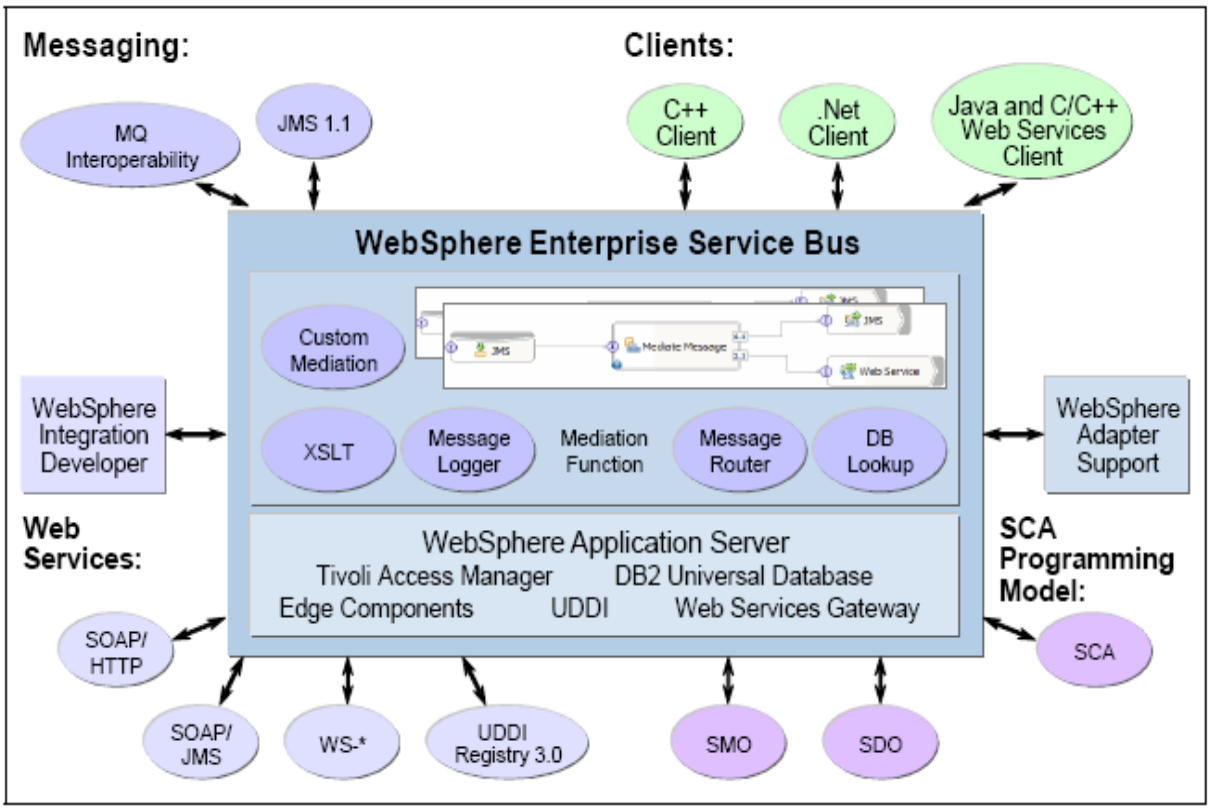


Figure 12 WebSphere ESB Product Overview

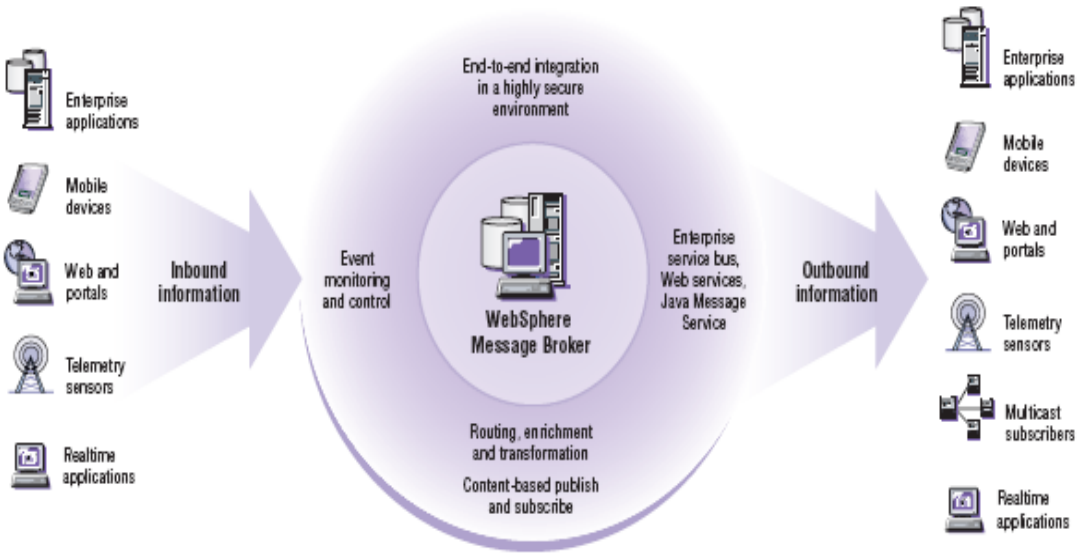


Figure 13 WebSphere Message Broker Overview

How does IBM's ESB's, WMB and WESB, improve the two big Enterprise Pressures – time-to-market and $$$?

WMB has a toolkit that acts as a very rapid development platform for creating message transformations from one format to another that might take a day of manual work but less than an hour with WMBT (the WMB development toolkit).

The following key benefits help automate the development of message "highways" within the company and to external partners for information and transaction exchange:

1. **Develop transformations that change your internal data such as (mainframe copy book) data to other LOB business data formats or external partner in 10% of the time of current manual methods.** This translates into substantial development cost savings and reduced time to market. Build components that transform messages from one data structure to another. For example, transform Cobol Copybooks to XML - Turns manual message transformation development time from days to hours with WMB toolkit.
2. **Rapid Visual re-routing to new business partners makes the enterprise more valuable because of its ability for quick and flexible business driven changes**: Rapid reconfiguration of routing from one business partner to new business partner in hours not days. For example, adding or removing hotel vendors for travel site.
3. **Publishing:** Business partners can subscribe to publishing of messages such as new price list.

Which Product to Choose?

Table 2 contains the key high level questions that can help you decide which is the best ESB for your situation. The answer might in fact be both. Likely you would start with one or the other. When these questions don't provide a decisive answer or detailed due diligence is called for, walk through Table 3 and Table 4.

Key Questions – ESB Selection			
	WESB V6	***WMB V6***	***Comments***
What supporting products are in the existing environment?	WAS	WMQ	Each uses the indicated product for its infrastructure. Leverage not only the product infrastructure and licensing but also the skill set of the team currently supporting each product
Is WPS (processer server) currently in or is part of the solution?	**High** Both use WID Both use WAS Both WPS and WESB use SCA components.	**Low** No interdependencies	WPS uses WAS for its base infrastructure The WID is used both for WESB and WPS for developing applications.

Key Questions – ESB Selection			
	WESB V6	***WMB V6***	***Comments***
Legacy integration required?	**Medium** JCA Adapters only	**High** Built-in CICS, VSAM, QSAM	WMB nodes allow high fidelity manipulation of legacy interactions
Strong Web Services Required?	**High** Latest Web Services standards is always in base WAS.	**Medium** No 2 PC transactions	
Extensive XML to non-XML data transformations required?	Medium	High	WMB has many nodes that simplify data mapping from one format to another. See table below for details.
Support for many data formats?	Medium	High	WMB has built in support for many data formats including HL7, EDI X12, EDIFACT and SWIFT. See table below for details.
Complex event processing	Med	High Nodes to process both	
You have a mixture of implementation languages?	Java	Java/C	
Strong Standards Environment required?	**High** SCA/SMO/SDO Web Services	**Medium** Web Services Node architecture is propriety	SCA strong push in industry for standardization. SDO is an industry standard.

Table 2 Key Questions when Selecting an ESB

IBM ESBs – Comparison of Critical Dimensions		
	WESB V6	WMB V6
Web Services Strength	**Sophisticated**	**Basic**
Scalability, Reliability & Performance	**Excellent** Deploy mediation modules to a WAS cluster provides failover and multiple mediation of request for improved throughput.	**Excellent** Multiple broker domains, brokers and execution groups
Legacy Interaction Strength	**Acceptable** (through Adapters)	**Extensive**
Product Maturity	**Medium** 2005 - 1st release V6/ Base technology WAS is rock solid	**High** Early 2000, Toolkit retooled circa 2004

IBM ESBs – Comparison of Critical Dimensions		
Unique benefits	**Web Services 2 PC capable.** **Can use fast J2C based Adapters that are the strategic direction for IBM connectors technology**. J2C connectors within WESB JVM, WBI are in own JVM.Fast connectors include JDBC, SAP, PeopleSoft, Siebel Binds with stateless Session EJB from service consumer	**Service request does not have to be in a specific format.** Handles many non-XML to XML and the reverse with built-in capacity or with SupportPacs. Many built-in message parsers.
Gotcha's	**Accepts SOAP, XML and JMS format only but can use external products to translate non-XML.** Must use WebSphere Transformation Extender (WTX), DataPower, J2C or WBI adapter to transform non-XML message to XML. Limited Data transformation to Web Services and J2EE messaging Standards	**No 2PC Web Services**. No WS-Transactions support. **No J2C Adapters.** Slower more complicated WBI adapter to connect to other Enterprise components – Siebel, PeopleSoft, SAP etc. **No WS-Security**
Technology	WebSphere Application Server V6	100% Native OS process
Building Blocks Technology	Mediation Modules SCA/SMO/SDO – Service Component Architecture is close to being an industry standard. SDO is currently an industry standard.	Message Flows and Message Sets Proprietary "Node" based components
Messaging Technology	Built-in WAS SIB V6	WMQ V6
Toolkit	WID – WebSphere Integration Developer	WMBT – WebSphere Message Broker Toolkit

Table 3 IBM ESB's - Comparisons of Critical Dimensions

ESB 401- Advanced ESB for Large Organizations

In a more complex organization (read large) neither WESB nor WMB alone will likely be able to solve all the requirements for an ESB. The likely scenario is both products will be combined to establish the logical concept of an ESB.

Figure 14 shows how WESB and WMB can be combined effectively to satisfy a request that require accessing both newer Web Services capable systems, Enterprise Systems such as SAP and legacy system such as a COBOL application. Note that the WESB would act as the gateway component responsible for exposing Web Services while WMB would likely channel the processing of complex legacy requests.

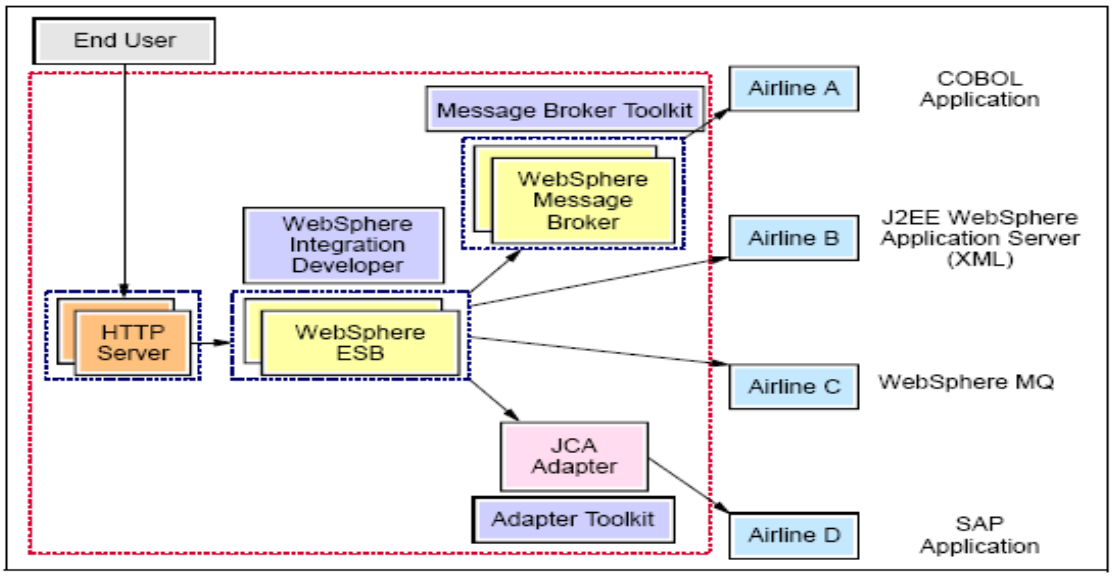


Figure 14 Large Organization ESB Solution - Both WESB and WMB

The scenario is:

1. A flight reservation system makes a Web Services requests with an XML[1] message.
2. The request is received by the HTTP Servers and is forwarded to WESB.
3. Routing capacity in the services application determines whether one or all the airlines are involved in the request.
4. WESB can convert some messages into the airline's format and forward requests.
    a. Airline A – XML[1] is sent unconverted to WMB and is processed in a message flow.
    b. Airline B – XML[1] is converted to XML[2] and sent over HTTP to the partner's system.
    c. Airline C – XML[1] is converted to XML[2] and sent over JMS.
    d. Airline D – XML is converted to a SAP formatted message and sent via JCA adapters.

References

1. Patterns: SOA Design using WebSphere Message Broker and WebSphere ESB, IBM Redbooks

Appendix

IBM ESB's – Detailed Dimensions Comparison			
Attribute	***WESB***	***WMB***	***Comments/Advantages***
Implicit Data Payload Transformations	**Limited to Web Services** Supports transformation of XML, SOAP JMS message data format (many more if used with adapters) Provides prebuilt mediations for XML transformation Custom transformation logic implemented in Java, XSLT	**Any to Any** <u>Transform</u> Self-defined Messages – XML Built-in Types – SOAP,MIME, others) Custom User-Built Messages – MRM – C, Cobol Copybook **Built-in** XML transformation and can import C structure, Cobol copybook, WSDL amongst other formats Custom transformation logic can be implemented in Java, ESQL, or XSLT. **<u>Additional Built-in Format Imports</u>** • Biztalk • TIBCO Rendezvous • Tuxedo • HIPAA • EDI-FACT • ACORD • ebXML • EDI-X.12 • AL3 • Word/Excel/PDF • HL7	
Protocol Transformation	Http, JMS, MQ , IIOP SMTP and FTP with adapters both in and out of the ESB	MQ,JMS, HTTP,EIS,FTP **TO** MQ,JMS,HTTP,CICS,FTP, IIOP,LDAP and many others	

		EIS (Mainframe) and FTP via node supportpacs	
Web Services Support	SOAP/HTTP(S) SOAP/JMS, WSDL 1.1 Supports WS-I Basic Profile 1.1 UDDI 3.0 Service Registry WS-Security WS-AtomicTransactions <u>Client support</u> Message Client for C/C++ and .NET Web Services Client J2EE Client	SOAP/HTTP(S) SOAP/JMS, WSDL 1.1 Supports WS-I Basic Profile 1.0 <u>Client support</u> JMS Message Client C/C++ & .NET Web Services Client MQI Client	Web Services through WMB can't participate in a 2PC Web Services transactions
Message Routing	Content and transport/protocol-based routing Supports through custom-built mediations using Java and the IBM SOA programming model (SCA and SDO) Provides prebuilt mediations for message routing	Content and transport/protocol based routing Custom routing logic can be implemented in Java or ESQL encapsulated in components called "Nodes"	
Connectivity			
MQ	MQ/JMS (via MQLINK configuration)	WMQ native Transport WMQ Mobile Transport (WMQ Everyplace client) WMQ Multicast Transport WMQ Real-time Transport W MQ Telemetry Transport/ MQTT	
JMS	JMS 1.1 (point-to-point, pub/sub)	JMS 1.1 (point-to-point, pub/sub) JMS Node supports input handling for virtually all third-party JMS systems including • WebLogic JMS	

		• SonicMQ JMS • TIBCO EMS JMS	
Http	http/https Built-in to WAS	MQ Web Services Node	
Mainframe	Indirectly through J2C adapters.	SupportPac nodes CICS (EXCI) Request, VSAM QSAM, flat-files	
Product Extensions	Binds with stateless Session EJB as a service consumer	**SupportPac** • SendMail Plug-in • LDAP Plug-in • SWIFT • FIX –Financial Exchange • FTP Server Input Node • TLOG Processor • Messsge Service client C/C++ • TCP/IP Sockets Node • Unzip Plug-in • XML Validator Node • Mercator • JText • WSRR Registry • SOAP Envelope Node • Inbound POP3 email/SMTP • Message Client for .NET	
Connectivity by Adapter Technology	**WBI and J2C adapters** **Strategic direction is J2C adapters.** **WebSphere Adapters based on J2C include** • JDEdwards EnterpriseOne • Oracle E-Business Suite • PeopleSoft Enterprise • SAP Software • Siebel Business	**WBI adapters** **No J2C adapters support** **WebSphere Business Integration include** • Ariba Buyer • I2 • JDEdwards EnterpriseOne • SAP Exchange Infrastructure	**J2C Adapters run in the same JVM as WESB which makes for tighter integration and higher performance. WBI run in a separate JVM.**

	Applications • DTS – Fujitsu Mainframe • Email – SMTP, POP3, IMAP • FTP • Flat File • JDBC •	• SAP mySAP • Siebel eBusiness Applications • SunGard FRONT ARENA • EJB's • CORBA • COM • Email – SMTP, POP3, IMAP • FTP • Exchange Server 2000 • JText (Flat File) • iSeries • Lotus Domino • TCP/IP Sockets • Data Handler for Complex objects – used for MS Word, PDF, HL7, Cobol Copybook • Data Handler for XML	
Message Logging	Pre built mediations for logging	Supports message logging	
Event Driven Processing	Supports event-driven processing by leverage adapters for capture and dissemination of business events	Supports complex event processing (processing of events formed by several earlier ones)	
Security	WS-Security Support HTTPS Authentication/Authorization via WAS J2EE	HTTPS Authentication/Authorization via Operating System	

Table 4 IBM ESB's – Detail Dimension Comparison of WESB and WMB

Below is the URL for WMB SupportPacs.

http://www-1.ibm.com/support/docview.wss?uid=swg27007197#5

The following table shows the nodes for the WMB product.

WMB Transport Support

Nodes	Transport
► JMSInput ► JMSOutput	*WebSphere Broker JMS Transport* is used to allow a message flow to receive messages from JMS destinations or to send messages to JMS destinations. These destinations are accessible through connection to a JMS provider. The JMS nodes work with the WebSphere MQ JMS provider, WebSphere Application Server Version 6.0, the service integration bus, and any JMS provider that conforms to the Java Message Service Specification Version 1.1.
► SCADAInput ► SCADAOutput	*WebSphere MQ Telemetry Transport* is a lightweight publish/subscribe protocol flowing over TCP/IP. This protocol is used by specialized applications on small footprint devices that require a low bandwidth communication, typically for remote data acquisition and process control.
► Real-timeInput ► Real-timeOptimizedFlow ► Publication	*WebSphere MQ Multicast Transport* is used by dedicated multicast-enabled JMS application clients to connect to brokers. Applications communicate with the broker by writing data directly to TCP/IP ports. This protocol is optimized for high volume, one-to-many publish/subscribe topologies.
► HTTPInput ► HTTPReply ► HTTPRequest ► Publication	*WebSphere MQ Web Services Transport* allows Web services clients using XML messages and the HTTP protocol running over TCP/IP to communicate with applications through message flows in a broker.
► Real-timeInput ► Real-timeOptimizedFlow ► Publication	*WebSphere MQ Real-time Transport* is a lightweight protocol optimized for use with non-persistent messaging. JMS applications communicate with the broker using TCP/IP ports.

Nodes	Transport
► MQInput ► MQOutput ► MQGet ► MQReply ► MQOptimizedFlow ► Publication	*WebSphere MQ Enterprise Transport* supports WebSphere MQ applications that connect to WebSphere Business Integration Message Broker by writing data to and reading data from message queues.
► MQeInput ► MQeOutput ► Publication	*WebSphere MQ Mobile Transport* is used exclusively by WebSphere MQ Everyplace clients. WebSphere MQ Everyplace is an application designed primarily for messaging to, from, and between pervasive devices. These are typically small, handheld devices, such as mobile phones and PDAs. A bridge queue on the broker's queue manager provides an interface for the WebSphere MQ Everyplace clients to the broker services.

WMB Message Manipulation

Node	Function
Nodes for message manipulation	
► Compute ► JavaCompute	Used to examine a message and create new messages. The Compute node logic is written in ESQL. The JavaCompute node uses logic.
► Database ► DataDelete ► DataInsert ► DataUpdate ► Warehouse	Used to interact with an ODBC datasource.
► Extract	Used to extract the exact contents of the input message that you want to be processed by later nodes in the message flow.
► Mapping	Uses the Mapping node to construct one or more new messages by creating new messages and populating them with new information, with modified information from the input message, or with information taken from a database.
► XMLTransformation	Applies a stylesheet to an XML message.

Node	Function
► JMSMQTransform ► MQJMSTransform	Used to send messages to older message flows and to interoperate with WebSphere MQ JMS and WebSphere Message Broker publish/subscribe.
► AggregateControl ► AggregateReply ► AggregateRequest	Used to combine the generation and fan-out of a number of related requests with the fan-in of the corresponding replies, and compile those replies into a single aggregated reply message.
Nodes for decision making	
► Check	Validates the format of a message.
► Filter ► JavaCompute	Routes a message based on conditional logic. The Filter node uses ESQL logic, while the JavaCompute node uses Java.
► FlowOrder	Used to control the order in which a message is processed by a message flow.
► Label, RouteToLabel	Uses the Label node in combination with a RouteToLabel node to dynamically determine the route a message takes through the message flow, based on its content.
► ResetContentDescriptor	Used to request that the message is reparsed by a different parser.
► TimeoutControl ► TimeoutNotification	Used together in a message flow for an application that requires events to occur at particular times, or at regular intervals.
► Validate	Used to ensure that the message is routed appropriately through the message flow.
Nodes for error handling	
► Throw	Used to throw an exception within a message flow.
► Trace	Used to generate trace records that can incorporate text, message content, and date and time information, to help you to monitor the behavior of the message flow.
► TryCatch	Used to provide a special handler for exception processing.