To ESB or Not to ESB?
Do you have to have an ESB to have a SOA?

Denise Hatzidakis
Chief Technologist
Perficient, Inc
denise.hatzidakis@perficient.com
denise5@us.ibm.com
We describe the enterprise service bus first and foremost as an architectural pattern. In fact, it is possible to construct service buses from a variety of different underlying integration technologies...

The architecture pattern remains valid and is a guiding principle to enable the integration and federation of multiple service bus instantiations.

Rob High, SOA Foundation Chief Architect, in SOA Foundation Architecture Whitepaper
What Is An Enterprise Service Bus?

- An ESB enables standards–based integration between loosely–coupled apps and services within and across
  - services oriented architectures - where distributed applications are composed of granular re–usable services with well–defined, published and standards–compliant interfaces
  - message driven architectures – where applications send messages through the ESB to receiving apps
  - event driven architectures – where applications generate and consume messages anonymously

- Mediations within an ESB enable intelligent processing of service request/responses, events, messages
  - at application endpoints or distributed through the infrastructure of the Bus
  - capabilities include:
    - transformations (e.g. XML to XML translations, DB lookups, aggregations), service selection, content–based routing
    - customized logging, metering & monitoring
    - autonomic behavior (e.g. system & business event detection, self configuration, healing, optimization, etc.)

- Enabling simple application integration across different platforms, programming models & messaging standards
  - underpinning Business Process and managed Business Partner integration
SOA Reference Architecture

Model of the Logical Architecture

Business Innovation & Optimization Services

Provide for better decision-making with real-time business information

Interaction Services

Enables collaboration between people, processes & information

Process Services

Orchestrates and automates business processes

Information Services

Manages diverse data and content in a unified manner

ESB

Enable inter-connectivity between services

Partner Services

Connect with trading partners

Business App Services

Build on a robust, scalable, and secure services environment

Access Services

Facilitate interactions with existing information and application assets

Infrastructure Services

Optimizes throughput, availability and performance

IT Service Management

Manage and secure services, applications & resources
Core Principles of the ESB Architectural Pattern

- ESB inter-connects requestor and provider
  - Interactions are *decoupled*
  - Supports key SOA principle – *separation of concerns*

- ESB provides *Service Virtualization* of
  - *Identity* via routing
  - *Interaction* via conversion
  - *Interface* via transformation

- ESB also enables *Aspect Oriented Connectivity*
  - Security
  - Management
  - Logging
  - Auditing
Virtual Provider Pattern – Implementation Considerations

Implementation Considerations

- Who builds the Virtual Provider?
  - The Service Consumer?
  - The Prospective Service Provider?
  - The Service Broker?
  - The Internal Enterprise-wide Governance Body on SOA?

- The Service Consumer may have to foot the bill or agree to split it with the prospective provider.
  - The provider may not be ready or willing (no funding or not a priority) to build a service interface for the consumer.
Enterprise Service Bus Architecture

- Protocol mapping
- Pattern recognition
- Service Brokering

**Micro Composition/Decomposition & Customized Routing**

- Message Validation
- Message Enrichment
- Database augmentation

**Enterprise Service Bus – Core Layer**

- Publish/Subscribe
- Assured Delivery
- High Availability

- Queuing
- Security
- Mediation Infrastructure

- Format Libraries
- Request/Response
- Management

**Bus Connections**

- Adaptors
- CICS
- IMS
- JMS

- .NET
- SOAP
- HTTP
- C/C++

- JCA/J2C
Key ESB Purpose Is Routing Service Requests

- The ESB provides the means to manage the service infrastructure and the capability to operate in today’s distributed, heterogeneous environment.
- A minimum capability ESB implementation includes:

<table>
<thead>
<tr>
<th>Communication</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing and addressing services providing location transparency</td>
<td>Support for multiple means of integration, for example, Java 2 Connectors, Web Services, and so on</td>
</tr>
<tr>
<td>An administration capability</td>
<td></td>
</tr>
<tr>
<td>At least one messaging paradigm (for example, request / response, pub/sub, and so on)</td>
<td></td>
</tr>
<tr>
<td>At least one transport protocol that is or can be made widely available</td>
<td></td>
</tr>
</tbody>
</table>

Service Interaction

Service messaging and interfacing model
## ESB Provides a Broad Set of Capabilities

<table>
<thead>
<tr>
<th><strong>Communications:</strong></th>
<th><strong>Service Interaction:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing, addressing, protocols, pub/sub, and async</td>
<td>Interface definition, service substitution, messaging model, SOAP, WSDL, and directories</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Integration:</strong></th>
<th><strong>Quality of Service:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Database, legacy, middleware connectivity, service aggregation, app server connectivity, and protocol transformation</td>
<td>Transactions and delivery assurance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Security:</strong></th>
<th><strong>Service Level:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication, authorization, non-repudiation, confidentiality, and standards support (WS-Security, Kerberos, and so on)</td>
<td>Performance, throughput, availability, and scalability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Message Processing:</strong></th>
<th><strong>Management and Autonomic:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoded logic, content-based logic, message and data transformations, and intermediaries</td>
<td>Service provisioning and registration, logging, metering, monitoring, systems management, and so on</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Modeling:</strong></th>
<th><strong>Infrastructure Intelligence:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Object modeling, common formats and libraries, public versus private, and so on</td>
<td>Business rules, policy driven behaviour, pattern recognition, and so on</td>
</tr>
</tbody>
</table>
Outside ESB

- Business Logic (Business Services)
  - ESB *does* contain integration logic or connectivity logic
  - Criteria: semantics versus syntax; aspects

Loosely coupled to ESB

- Security and Management
  - Policy Decision Point outside the ESB
  - ESB can be Policy Enforcement Point

Tightly coupled to ESB

- Service Registry
  - Registry a Policy Decision Point for ESB
  - ESB a Policy Enforcement Point for Registry
  - But, Registry has a broader scope in SOA

Tooling required for ESB

- Development
- Administration
- Configures ESB *via* Service Registry
Expanded View of the ESB

- Communication Protocols
  - Supply basic connectivity to requesters and providers
    - Impact QoS (e.g., reliable delivery, transactions)
  - Supply inherent Interaction Patterns (e.g., request/reply, one-way, pub/sub)
- An ESB leverages underlying communication fabrics of SOA infrastructure
  - ESB provides on-ramps and off-ramps
- Standards are important

<table>
<thead>
<tr>
<th>Typical requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP (SOAP/HTTP, XML/HTTP)</td>
</tr>
<tr>
<td>MQ (SOAP/JMS/MQ, XML/MQ, text/MQ, ...)</td>
</tr>
<tr>
<td>Adapters (legacy, EIS)</td>
</tr>
<tr>
<td>WS-I, WS-Security</td>
</tr>
<tr>
<td>RAMP</td>
</tr>
</tbody>
</table>
ESB Transport Protocols and Conversion

- Basic connectivity supported via one or more transport protocols
  - Dependent on underlying communication fabric(s)
- Conversion inherent with support for more than one transport protocol
- Enables
  - Virtualization of interaction protocol
  - Aspects of QoS (e.g., reliable delivery, transactions)
- Typical requirements
  - HTTP (SOAP/HTTP, XML/HTTP)
  - MQ (SOAP/JMS/MQ, XML/MQ, text/MQ, ...)
  - Adapters (legacy, EIS)
- Standards important
  - WS-I
  - ...
**Expanded View of the ESB**

- **Message Models**
  - Describe messages exchanged with requesters and providers
  - Based on Meta-models
    - Fundamental means of describing messages
    - For example, XML Schema language
  - Content models define specific messages
    - For example, XML schema

- An ESB supports one or more message meta-models
- An ESB supports multiple message content models
  - Can include industry standard models as well as enterprise specific models
  - Can include weakly-typed models

**Typical requirements**
- XML schema definition
- Industry specific content models
ESB Message Models and Transformation

- An ESB supports multiple message models
  - Configurable for specific internal message models
  - Canonical data model(s) optional

- Should provide transformation between message models
  - Via mediation pattern
  - Via adapters

- Enables
  - Virtualization of *interface*
  - Aspects of QoS (*e.g.*, performance)

- Typical requirements
  - Industry specific
  - Enterprise specific
Mediation Flows

- Process messages exchanged between requester and provider via ESB
  - Large grained
  - Moderately reusable
  - Constructed from Mediation Patterns
- Mediation Patterns define processing “steps” of a mediation flow
  - Small to middle grained
  - Highly reusable
  - ESB products include pre-built “mediation primitives”
ESB Mediation Patterns and Message Processing

- Allow manipulation of messages during a message flow
  - Provided by a mediation framework enabling pattern construction
- Enhance the basic interaction patterns, *e.g.*,  
  - Message enrichment
  - Monitoring and logging
  - Registry, security and management
  - Distribution/aggregation

- Enables
  - Aspects of QoS (security and management)
- Typical requirements beyond routing and transformation
  - Retry
  - Recipient list
  - Custom
ESB Mediation Flows and Mediation Patterns

- Mediation enables **Service Virtualization** of
  - **Identity** via routing
    - Using basic mediation patterns (context, content, contract)
    - Using composed mediation patterns (retry, failover, Distribution/aggregation, ...)
    - Dynamic, driven by metadata in registry
    - Impact aspects of QoS (e.g., SLA, failover)
  - **Interaction** (of protocol and pattern) via conversion
    - Protocol conversion inherent with support for more than one transport protocol
    - Impact aspects of QoS (e.g., reliable delivery, transactions)
  - **Interface** via transformation
    - Using specific mediation patterns
    - Using adapters
    - Impact aspects of QoS (e.g., performance)
    - NOTE: other forms of mediation should be agnostic to interface (weakly-typed processing)

- Mediation enables **Aspect Oriented Connectivity**
  - Security & Management
  - Logging
  - Auditing
  - ...

- Provided by a mediation framework
  - Offering pre-built mediation pattern (primitives) support
  - Enabling mediation pattern composition

Typical requirements
- Dynamic routing
- Logging
Mediation Patterns – Examples

- Request / Response
- Request / Multi Response
- Event Propagation
- Protocol Switch
- Transform
- Enrich
- Route
- Distribute
- Monitor
- Correlate
- Canonical Adapter
- Transform – Log – Route
- Gateway
ESB Interaction Patterns and Enhanced Routing

- **Fundamental interaction patterns based on underlying communications fabric(s)**
  - Point-to-point
    - Request/reply (synchronous and asynchronous)
    - One way
  - Pub/Sub

- **Enhanced (dynamic) routing of messages**
  - via mediation patterns

- **Enables**
  - Virtualization of location and identity
  - Aspects of QoS (e.g., SLA, failover)

- **Typical routing requirements**
  - Round robin
  - Content based
  - Service registry driven
What Is Not in the Bus

- In the ESB – capability necessary to support service virtualization
- Not in the ESB – everything else!
  - But things not in the ESB may use or be used by the ESB
Why Loose Coupling?

- Tighter coupling tends to cost more over time:
  - Synchronizing multiple organizations on change
  - Adapting, redeploying updated components without affecting others
  - Making changes is hard and expensive, or impossible:
    - Knowledge is distributed throughout the code
    - Same people are solving business and infrastructure problems
  - Different parts of the solution are difficult to manage separately
  - Hard to move, hard to scale, hard to distribute, hard to replace
  - More coupling implies more expensive testing

- Looser coupling requires greater investment up front:
  - More design work
  - More implementation work

Several service elements must be considered when thinking about coupling:

- Service
- Message
- Interface
- Contract
- Policy
- Conversation
- State
- Transactions
- Process
Loose Coupling Aspects of Service Interactions

- Consumer
- Provider
- Location
- Language
- Data Format
- Delivery Assurance
- Semantic Interface
- Interaction State
- Business Data Model
- Platform
- Protocol
- Security
- Service Version
- Service Provider Identity
- Time
- Service
Loose Coupling:
“Loose” Is Not a Well Defined Term

- **Coupled**
  - Directly manipulated by service requester and provider application code.
  - *e.g.* Business data model

- **Declared**
  - Specified in the service interface but not manipulated in application code
    - *e.g.* Transactional behaviour declared in interface
  - Transformed
    - Specified in interface, manipulated in code
    - Infrastructure mediates to accommodate change
    - *e.g.* different data formats declared in interface, transformed by infrastructure
  - Negotiated
    - Requester and Producer offer a spectrum of capabilities
    - Infrastructure negotiates an agreed behaviour
    - *e.g.* Combination of certificates and encryption strength, via WS-Policy

- **Decoupled**
  - Changes by one party do not require changes by the other
  - *e.g.* service location, infrastructure manages routing
# Coupling Details

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Meaning</th>
<th>Coupling Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic Interface</strong></td>
<td>The structure and meaning or intent of business actions and the data associated with them.</td>
<td>Coupled – within the service design principles concerning granularity, choreography, connections etc.</td>
</tr>
<tr>
<td><strong>Language and Platform</strong></td>
<td>The programming models and languages (e.g. J2EE, Visual Basic, BPEL) used to implement service consumers and providers and the operating and application environments supporting them (e.g. WebSphere, Windows, Tuxedo, IMS).</td>
<td>Decoupled</td>
</tr>
<tr>
<td><strong>Data Format</strong></td>
<td>The precise structure (e.g. relationships, hierarchies, containments) and formats (e.g. US vs. European dates) of data, and electronic formats recording it (e.g. XML, copybooks, code pages, comma-delimited, encrypted, compressed), etc.</td>
<td>Decoupled or Transformed</td>
</tr>
<tr>
<td><strong>Protocol and Location</strong></td>
<td>The protocol through which service requests are invoked and accepted (e.g. HTTP, WebSphere MQ, FTP, RMI/IIOP) and the address which is used to identify it (e.g. URL, JNDI, etc.).</td>
<td>Decoupled, Transformed or Negotiated</td>
</tr>
<tr>
<td><strong>Service Provider Identity</strong></td>
<td>Some services may have multiple suppliers (e.g. when accessing multiple insurance quotes). Others may hide the identity of the true supplier (e.g. when you buy insurance from a broker, or supermarket).</td>
<td>Declared or Negotiated</td>
</tr>
<tr>
<td><strong>Service Provider Implementation</strong></td>
<td>The service supplier may change the implementation of their service (e.g. by replacing a legacy system).</td>
<td>Decoupled</td>
</tr>
</tbody>
</table>
| **Time**                    | - Are consumers and providers active at the same time? Is this true in the case of asynchronous request / response, publish / subscribe or event-based models? Should unplanned availability issues be taken into account?  
- As IT systems show many differing planned and unplanned availability characteristics (such as 24/7 verses working hours), service interactions will sometimes need to span systems with different characteristics. | Declared or Transformed                                                         |
## Coupling Details

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Meaning</th>
<th>Coupling Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delivery Assurance</strong></td>
<td>In any interaction there is a finite risk that information will be lost. This risk should be balanced against the value of the data associated in the interaction, and appropriate safeguards put in place to control the risk of failure. This question becomes more complex in SOA where service consumers expect to be offered a clear definition of service levels, but where service fulfillment might involve many intermediaries (brokers, infrastructure, etc.) and associated technologies between the consumer and the actual service provider.</td>
<td>Declared or Negotiated</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>As with any other interaction, standard security concerns must be addressed such as identification, authentication, authorization, confidentiality, integrity and non-repudiation.</td>
<td>Declared, Negotiated or Transformed</td>
</tr>
<tr>
<td><strong>Service Version</strong></td>
<td>As with any other interaction, changes will be required over time to reflect changing requirements or fixing bugs. In a distributed infrastructure of service interactions, it is unlikely that all participants will be able to upgrade at the same time. Similarly, the use of large-grained interfaces may make some participants tolerant to certain changes in service interfaces, particularly the addition of optional attributes to the data model.</td>
<td>Declared or Negotiated</td>
</tr>
<tr>
<td><strong>Interaction State</strong></td>
<td>Many services will either be invoked as part of an ongoing process consisting of other services, or be invoked to act on business data that is expected by the service consumer to be in a particular state. Loosely coupled integration will only be achieved if stateful relationships are modeled in a careful and explicit manner.</td>
<td>Declared</td>
</tr>
</tbody>
</table>
Semantic Interface

Meaning
- The structure and meaning or intent of business actions and the data associated with them.

- Coupling Intent
  - Coupled – within the service design principles concerning granularity, choreography, connections, etc.

- Techniques
  - This topic was discussed in the lectures on service design, identification and modelling.
  - Business systems must share an understanding of the tasks and data processed by the service.
  - Shared business object libraries or XML schemas can be exploited.
  - Future semantic technologies such as Resource Definition Framework [RDF] and the Web Ontology Language [OWL] (see the www consortium at http://www.w3c.org)
  - Cross-industry or vertical XML or Web Services standards
  - SOA interface definitions
  - Process definitions, e.g. BPEL
  - Transformations, aggregations, choreographies or enrichments of data can be performed by aggregation or choreography.
Language & Platform

- **Meaning**
  - The programming models and languages (*e.g.* J2EE, Visual Basic, BPEL) used to implement service consumers and providers and the operating and application environments supporting them (*e.g.* WebSphere, Windows, Tuxedo, IMS)

- **Coupling Intent**
  - Decoupled

- **Techniques**
  - Language and platform-independent interface definition, *e.g.* IDL, WSDL, XSD
  - Language and platform-independent data formats, *e.g.* XML
  - Language and platform-independent communication protocols, *e.g.* IIOP, SOAP, WebSphere MQ.
  - Invocation APIs (*e.g.* WSIF, JAX-RPC),
  - Adaptors or ESB / EAI infrastructure to integrate applications to the interface definitions and data formats. Protocol transformation. Integration and client connectivity function.
Data Format

Meaning

- The precise structure (e.g. relationships, hierarchies, containments) and formats (e.g. US vs. European dates) of data, and electronic formats recording it (e.g. XML, copybooks, code pages, comma-delimited, encrypted, compressed), etc.

Coupling Intent

- Declared or Transformed

Techniques

- Language and platform-independent data formats such as XML.
- Adapters, XSL style sheets, or bus infrastructure required to support transformations between data formats, such as between COBOL copybooks and XML.
- Application development tool wizards can create language-specific representations of some data formats, particularly XML.
- Other aspects of data format that are critical to real world SOA implementations are data encoding, code pages and data compression, including XML compression techniques.
Protocol and Location

- **Meaning**
  - The protocol through which service requests are invoked and accepted (*e.g.* HTTP, WebSphere MQ, FTP, RMI/IIOP) and the address which is used to identify it (*e.g.* URL, JNDI, *etc.*).

- **Coupling Intent**
  - Declared, Transformed or Negotiated

- **Techniques**
  - Service invocation mechanisms for service requesters and providers that does not specify service protocol or locations, for example, an implementation of JAX-RPC with support for multiple protocols.
  - Adapters or ESB infrastructure can perform service routing and protocol transformation, and might identify routing and protocol choices dynamically by applying policies at runtime.
  - Service directories and communication routing tables provide location virtualization.
Service Provider Identity

- Meaning
  - Some services may have multiple suppliers (*e.g.* when accessing multiple insurance quotes). Others may hide the identity of the true supplier (*e.g.* when you buy insurance from a broker, or supermarket).

- Coupling Intent
  - Declared or Negotiated

- Techniques
  - Service invocation mechanisms that enable service substitution, for example JAX-RPC.
  - Adapters or ESB infrastructure can perform service routing to different providers.
  - Directory (for example, UDDI) or broker intermediary to decide who fulfills the service each time.
  - An ESB might identify a suitable service provider based on WS-Policy, for example by selecting the cheapest or most-responsive provider available at the time.
  - Workload management technology in communication layers.
Service Provider Implementation

- Meaning
  - The service supplier may change the implementation of their service (e.g. by replacing a legacy system).

- Coupling Intent
  - Decoupled

- Techniques
  - Really a combination of the other aspects, but...
  - Service invocation mechanisms that enable service substitution, for example JAX-RPC.
  - Adapters or ESB infrastructure can perform service routing to different providers.
  - Directory (for example, UDDI) or broker intermediary to decide who fulfills the service each time.
  - An ESB might identify a suitable service provider based on WS-Policy, for example by selecting the cheapest or most-responsive provider available at the time.
  - Workload management technology in communication layers.
Time

- Meaning
  - Are consumers and providers active at the same time? Is this true in the case of asynchronous request / response, publish / subscribe or event-based models? Should unplanned availability issues be taken into account?
  - As IT systems show many differing planned and unplanned availability characteristics (such as 24/7 versus working hours), service interactions will sometimes need to span systems with different characteristics.

- Coupling Intent
  - Declared or Transformed

- Techniques
  - Use of asynchronous transport protocols, for example WebSphere MQ, WS-ReliableMessaging.
  - Use of an ESB or intermediary store and forward capability for asynchronous request / response, message correlation, and so forth.
  - Message correlation and transaction identifiers used to associate individual service interactions with longer ongoing business process interactions.
Delivery Assurance

Meaning
- In any interaction there is a finite risk that information will be lost. This risk should be balanced against the value of the data associated in the interaction, and appropriate safeguards put in place to control the risk of failure. This question becomes more complex in SOA where service consumers expect to be offered a clear definition of service levels, but where service fulfillment might involve many intermediaries (brokers, infrastructure, etc.) and associated technologies between the consumer and the actual service provider.

Coupling Intent
- Declared or Negotiated

Techniques
- Assured delivery communication protocols, for example WebSphere MQ, WS-ReliableMessaging.
- Error and exception handling processes, for example for SOAP faults, manual correction interfaces, exception reports.
- Use the features and deployment descriptors of containers, such as J2EE, in service implementations.
- Advanced WS standards, for example WS-ReliableMessaging and WS-Transaction.
- Negotiated through WS-Policy by the ESB.
- In order to provide a consistent end-to-end approach to delivery assurance, integrity and error handling for a chain of service interactions, it will often be necessary to combine several techniques used for individual interactions. These techniques might include handling communication failures, the use of synchronous two-phase commit, the ability to handle duplicate messages, and compensation schemes.
Security

- **Meaning**
  - As with any other interaction, standard security concerns must be addressed such as identification, authentication, authorization, confidentiality, integrity and non-repudiation.

- **Coupling Intent**
  - Declared or Negotiated or Transformed

- **Techniques**
  - Declared by WS-Security or negotiated through WS-Policy.
  - Point-to-point or communication-based security and trust models.
  - Implemented through applications or through third-party or intermediary components in the SOA architecture.
Service Version

Meaning
- As with any other interaction, changes will be required over time to reflect changing requirements or fixing bugs. In a distributed infrastructure of service interactions, it is unlikely that all participants will be able to upgrade at the same time. Similarly, the use of large-grained interfaces may make some participants tolerant to certain changes in service interfaces, particularly the addition of optional attributes to the data model.

Coupling Intent
- Declared or Negotiated

Techniques
- Service naming standards.
- Version-based routing in the bus infrastructure.
- Service request /provider tolerance of changes in optional data attributes.
- Published policies for concurrent support of multiple service versions and deprecation / retirement schedules.
Interaction State

- **Meaning**
  - Many services will either be invoked as part of an ongoing process consisting of other services, or be invoked to act on business data that is expected by the service consumer to be in a particular state. Loosely coupled integration will only be achieved if stateful relationships are modeled in a careful and explicit manner.

- **Coupling Intent**
  - Declared

- **Techniques**
  - Matching of messages or events to long-lived processes by explicit process or transaction IDs in semantic interface, or by application data (for example, customer ID).
  - Service Choreography or Workflow technology may provide some facility to use a variety of input data to associate messages with specific instances of processes.
  - Primary key matching technology such as provided by WebSphere InterChange Server.
  - The emerging WS-ResourceFramework provides a standard model for associating services with stateful resources.
  - Enterprise Application Integration middleware support for message aggregation and correlation.
  - Customized solutions involving custom message headers and stateful stores.
ESB Presents and Maps Services

- Facade Services
- Transformations and Aggregation
- Invokable Services

Enterprise

Service Requestors
- A
- B
- C

Inbound Port 1
Inbound Port 2
Inbound Port 3

Hub

ESB

Zone: ESB

ESB Namespace Directory

Administration Services

Service Providers
- D
- E
- F

Outbound Port 1
Outbound Port 2
Outbound Port 3
Enterprise Service Bus – An Overview

This is the Service Requestor-specific invocation protocol.

An ESB Namespace name is used over these connections.

The ESB Namespace name has been mapped to an outbound port address.

The outbound port maps the message to meet the ServiceProvider's Requirements.

This maps all the ESB Namespace names to their endpoints.

This is a single administration infrastructure.
ESB Helps Integrate Requestors and Providers

- API
- Protocol
- Adaptor

Enterprise

Service Requestors
- A
- B
- C

Inbound Port 1
Inbound Port 2
Inbound Port 3

Hub

Outbound Port 1
Outbound Port 2
Outbound Port 3

Service Providers
- D
- E
- F

ESB
- Zone: ESB
  - ESB Namespace Directory
  - Administration Services
ESB Service Interaction Characteristics

- Semantic Interface
- Language
- Platform
- Data Format
- Protocol
- Location
- Service Provider Identity or Implementation
- Time
- Delivery Assurance and Error Handling
- Security
- Service Version
- Interaction State
Service Implementations Can Easily Be Substituted

- Facade Service Unchanged
- Mapping Changed
- Service Substituted
ESB Can Be Physically Distributed

Distributed Infrastructure

Service Client
Service Client
Service Client

<Bus>ESB

Service Provider
Service Provider
Service Provider

Single Point of Control

Service Client
Service Client
Service Client

<Hub>ESB

Service Provider
Service Provider
Service Provider

Service Provider
Service Provider
Service Provider

<Hub> Runtime Node
<Hub> Runtime Node
<Hub> Runtime Node
<Hub> Runtime Node

Configuration and Control Services

Configuration

Control Client

Enterprise Service Bus
ESB Roles – Infrastructure Patterns

**ESB Focus:** rich service virtualization

**ESB Gateway Focus:** controlled and secure service interaction between internal or external domain boundaries

**Key context Issues:**
- Security
- Quality of Service
- Management
- Transactions
ESB Roles –
ESB Integration Topology Patterns

Direct

Multiple namespaces, administration domains; namespace mapping in each ESB; services are likely to be applicable throughout the enterprise

Brokered

Multiple namespaces, administration domains; namespace mapping in Brokered ESB Gateway facilitates service interaction; subset of services applicable throughout the enterprise

Federated

Multiple namespaces, administration domains; namespace mapping in Federated ESB facilitates service interaction with multiple implementations; subset of services applicable throughout the enterprise
ESB Roles – Adapter Patterns

- Adapters may be needed to connect systems and applications to the ESB
- Adapters types
  - Technology – XML, EJB, ...
  - Application – SAP, PeopleSoft, ...
  - Legacy – CICS, IMS, ...
- Adapters patterns (classified by governance domain)
  - EIS
  - Independent
  - Split
  - ESB
The What and the How of ESB

- **What functionality does the ESB support**
  - Service Virtualization
  - Communication Protocols
  - Data Models
  - Message Flows
  - Meditation Framework
  - Access to existing Enterprise Information Systems
  - Qualities of Service
  - Use of security, monitoring, and registry

- **How do the ESB products deliver the What**
  - Pre-built mediation primitives
  - Mediation programming model
    - Message data model
  - Policy/meta-data driven
  - Tooling support
  - Administration support
    - dynamic configuration
Enterprise Service Bus Pattern

**Drivers**

- Support large numbers of service interactions in a manageable way
- Provide support for advanced service interaction capability, for example, transactions, store and forward, infrastructure services, security, and quality of service
- Support a variety of interaction styles such as synchronous request/response, messaging, publish/subscribe, and events
- Support service routing and substitution, protocol transformations, and other message processing
- Support both Web Services and traditional EAI communication standards and technologies

**Implementations**

- EAI / messaging technology
- ESB products
- "Gateway" technology
Decisions...
ESB Considerations – Standards Supported

- **Technology**
  - Base specs: XML, SOAP, WSDL
  - From the Java community: J2EE, JMS, JAX-RPC/JAX-WS, JAX-B, SAAJ, SDO, (SCA).

- **Industry**
  - Open Application Group Integration Specification (OAGIS)
  - Health Level Seven (HL7)
  - Standards for Automotive Retail (STAR)
  - *etc.*

- **Corporate**
  - Coding conventions
  - Technology decisions
Key ESB Considerations

- **Transport Protocols and conversion**
  - Breadth of protocols supported on the bus
  - Support for specific JMS providers

- **Message Models and transformation**
  - Basic (SOAP) web services
  - Advanced web services support (e.g., WS-Security)
  - Message models beyond SOAP and XML
  - Industry standard message models
  - Un-typed message processing
  - XSLT
  - Any-to-any transformation
  - Adapters for legacy and EIS systems

- **Interaction patterns and enhanced routing**
  - Synchronous request/response and one-way only
  - Asynchronous messaging
  - Pub/sub
  - On-demand (deployment time or runtime) configuration or routing of messages

Note: this list is NOT all inclusive, it represents some key criteria which impact product selection
Key ESB Considerations

- **Mediation patterns and message processing**
  - Additional message processing, *e.g.*, enrichment
  - Complex interaction patterns, *e.g.*, aggregation
  - Extensiveness of pre-built mediation palette
  - Custom Mediation programming model

- **QoS**
  - Performance requirements
  - Guaranteed delivery
  - Heterogeneous transaction coordination

- **Non-functional requirements**
  - Existing and required skill set (e.g., J2EE skills)
  - Existing IT environment (e.g., J2EE application server)
  - Ease of integration with management (e.g., monitoring, security) environment
  - Product maturity and comfort level with leading edge products
  - Tooling affinity to current tool set
  - Price and total cost of ownership
  - Migration from and coexistence with existing messaging infrastructure
Case Studies
Case 1 Description

- **Customer environment**
  - This customer is mainstream adopter of technology. Views this as first step in building an ESB.

- **Business Requirements**
  - Standards based requestors/providers will use SOAP/HTTP
  - Be able to change provider implementation and physical deployment without impacting consumers
  - Support 50ms response time with moderate load
  - SOA security (*e.g.*, WS-Security, XML firewall, ...) for interaction with external requestors

- **Technical Requirements**
  - Many services deployed will require the same mediation flow – minimize administration and maximize ease of making new services available on the bus

- **Architecture Decisions**
  - Weakly typed mediation interfaces to support common message flows across many port types
Case 2

- **Customer environment**
  - This customer is leading adopter of technology. Comfortable with sophisticated SOA solutions.

- **Business Requirements**
  - Any provider must be accessible via multiple heterogeneous requestors
  - Support moderate volume of requests
  - Intranet environment does not require SOA Security or other complex security considerations
  - Global transactions across multiple heterogeneous transaction managers

- **Technical Requirements**
  - ESB must support
    - Communication protocol conversion, but not adapters
    - Flexible data model conversion, with acceptable performance and adequate tooling
  - Enterprise class persistent messaging backbone

- **Architecture Decisions**
  - Canonical data model(s) used in ESB
  - Up to the consumers and providers to adapt to the service definition supported by the ESB
Case 2: Multi-protocol Exchange –
Intermediary decoupling heterogeneous consumers and suppliers

Domain of interest - Intranet Exchange

WebSphere (WAS/Portal) - SOAP/JMS
.NET Client - SOAP/HTTP
Some Client - XML/HTTP
XML/MQ Client - XML/MQ
Text/MQ Client

ESB

Tooling

WebSphere provider
.NET provider
Some provider
CICS

SOAP/JMS
SOAP/HTTP
XML/HTTP
XML/MQ
COBOL Copybook/MQ

Some Client

WebSphere

Tooling
Case 3 Description

- **Customer environment**
  - This customer is leading adopter of technology. Comfortable with sophisticated SOA solutions.
  - WebSphere Application Server customer

- **Business Requirements**
  - The customer wants to provide web service access to functionality in an Enterprise Information System such as SAP R/3, PeopleSoft, or Oracle Financials.
  - Intranet environment does not require WS-Security or other complex security considerations
  - The integration is based on message exchange/data replication scenarios – there is no business process or data synchronization between clients and EIS systems
  - Support moderate volume of requests

- **Technical Requirements**
  - The targeted integration is point-to-point, although multiple EISs can be exposed as web services at the same time.
  - Data transformation should use XSLT; tooling important
  - Log the messages as they flow through the hub – want to log asynchronously to a file

- **Architecture Decisions**
  - Affinity to J2EE
  - Use available adapter product to simplify development
Case 3 – Adapting between web service requestor and an EIS system

Domain of interest - Intranet

Web Services requestor

Tooling

SAP Adapter

ESB

EIS Backend