JBossESB

SOA everywhere!
Aims

- To provide the standard OSS infrastructure for SOA
  - SOA principles first and foremost
- Use SOA principles *internally* as well as *externally*
  - Everything will (conceptually) be considered as a service
  - Everything will be replaceable
- Standards compliant
  - Though requirements live longer
JBossESB = SOA Infrastructure
JBossESB will provide

- Process orchestration
- Protocol translation
- Adapters
- Repositories (e.g., UDDI)
- Change management (hot deployment, versioning, lifecycle management)
- Quality of service (transactions, failover)
- Qualify of protection (message encryption, security)
- Management (versioning of services)
Requirements

- Cannot mandate specific capability implementations
- All capabilities accessed as services
  - Plug-and-play
  - Extensibility
- All capabilities are message based
  - Including (conceptually) the container
- Standards are important
  - JBI
  - Perhaps SCA
JBossESB and standards
Core message

• Everything is a service, including the bus
  • All services are interacted with via messages.
  • Includes service lifecycles
    ✓ Containers abstracted within architecture
    ✓ Services plugged directly into a lifecycle bus
  • Services can be plugged into multiple buses concurrently
The SOA Bus

- Underlying the ESB is a MOM abstraction
- Does not mandate implementation
  - JMS, SOAP etc.
    - ESB must be able to support pure-play Web Services deployments
    - WS-RX
- Capabilities can be provided by multiple implementations
  - Concurrently
- Support multiple buses
  - Single bus concept is wrong
    - Counter to SOA and Web Services
    - Biggest problem with old-style EAI
ESB versus SOA versus EDA

- SOA rules take precedence
  - EDA is a way of implementing SOA
    - JBossESB is a unitary EDA
- ESB is a narrowing of SOA
  - Mediation not necessary for SOA
  - Routing not necessary for SOA
- SOA infrastructure first and foremost
  - ESB veneer
Message view
Core service requirements

- Container
  - JBoss Microcontainer default
- MOM
  - JBoss Messaging and Web Services
- Transformations
  - JBoss Rules
- Repository
  - UDDI
  - Basic contract definition
    - QoS
    - Service versions
Standards

• Important for interoperability
  ✓ Web Services
  ✓ JMS

• Important for portability
  ✓ JBI
  ✓ SCA

• But
  ✓ Requirements are more important
  • Standards change!
JBossESB interfaces
Addressing

• Logical and physical names
  ✓ Logical requires indirection to lookup

• WS-Addressing based
  ✓ Not dependant on Web Services
  ✓ Just capabilities
The Message

- Two levels of message
  - ✔ Seen and used by clients and services
  - ✔ Seen and used by the core ESB
- Latter is a superset of the former

```java
interface Message {
    public Header getHeader ();
    public Context getContext ();
    public Body getBody ();
    public Fault getFault ();
    public Attachment getAttachment ();
}
```
interface ClientPlugin
{
    public void send (Address to, Body msg);
    public void sendAsync (Address to, Body msg);
    public void sendAsync (Address to, Body msg, Callback cb);
    public void sendReliable (Address to, Body msg);
}

interface ClientPluginFactory
{
    public ClientPlugin getPlugin (ContractDefinition def);
}
Service plugin

interface ServicePlugin
{
    public Body receive (Address from);
}

interface ServicePluginFactory
{
    public ServicePlugin getPlugin (ContractDefinition def);
}
But ...

- Lots to do
- We have many components
- We need to collaborate with partners
  - SOA-within-and-without should help
  - Best-of-breed approaches to ESB deployments
    - No single solution
  - Talking to partners and vendors now
- JBossESB as the unifying infrastructure
JBossESB Needs You!