JBI and OpenESB
Introduction to Technology

Michael Czapski
Advanced Solutions Architect, SOA/BI/Java CAPS
Sun Microsystems, ANZ
Learn what JBI and OpenESB are intended to address and how they go about it.
What is the problem?

- Proprietary Enterprise Application Integration Technologies
- Enterprise Application Integration vendor lock-in
Agenda

- EAI
- JBI
- OpenESB
- Demonstration
Agenda

- EAI
- JBI
- OpenESB
- Demonstration
Enterprise Application Integration

Uses of software and computer systems' architectural principles to integrate a set of enterprise computer applications


Purposes

- Data/Information Integration
- Process Integration
- Common Facade
- Legacy De-coupling
- Foundation for Implementation of Enterprise Architectures
- Vendor Independence
Enterprise Application Integration

- Approaches
  - File Transfer
  - Shared Database
  - RPC
  - Messaging

- Models
  - Point-to-Point
  - Hub-and-Spokes / Brokered
  - Bus
Enterprise Application Integration

Services

• Connectivity
• Message Routing
• Data Transformation
• Data Enrichment
• Data Replication
• Orchestration
• Composition
• Mediation
• Federation
• Auditing
Agenda

- EAI
- JBI
- OpenESB
- Demonstration
What is JBI?

- JCP JSR (208) defines an extensible, standards based integration architecture
- Allows third-party components to be “plugged in” into a standard integration infrastructure
- Components communicate via WSDL-based mediated message exchanges
- Defines a 'meta-container' or 'container of containers'
- Dependency on Java SE/Java EE

  - jcp.org/aboutJava/communityprocess/final/jsr208/index.html
Who is participating in JBI?

- Apache
- Borland
- Cap Gemini
- Collaxa
- DPWN
- Fujitsu
- Intalio
- IONA
- IOPSIS
- JBoss
- Nokia
- Novell
- Oak Grove Systems
- Oracle
- Research In Motion
- SAP AG
- Sonic Software
- Sun Microsystems
- Sybase
- Tmax Soft
- TIBCO
- Vignette
- WebMethods
- Sid Askary
- Jie Liu
- Timothy Potter
Why JBI?

- Provides an open integration infrastructure
- Provides a service-oriented integration framework
- Creates a multi-vendor pool of integration components
- Give users the choice of components to use
- Eliminates vendor lock-in
- Fosters innovation
What does JBI provide?

- Standard “meta-container” for integrated services
- Manages communication between components
- Hosts plug-in:
  - Service Engines: business logic
  - Binding Components: communications protocols
- Standardises component interfaces
- Standardises component lifecycle
- Standardises deployment model
- Standardised monitoring and management framework
- Fosters loose coupling via WSDL interfaces
- Implements WSDL 2.0 Message Exchange Patterns
What does JBI Architecture look like?

- Service Engines
- Normalized Message Router
- Binding Components

J2SE™/J2EE™ Platforms

JBI Core Services
- Installation
- Deployment
- Management

System Management
Example SEs and BCs

- Orchestration (BPEL)
- Transformation (XSLT)
- JavaEE Platform
- Normalized Message Router
- HTTP
- FTP
- AS2
- JMS

Service Engines
Binding Components

J2EE™ Platform
For whom is JBI intended

- Developers of Service Engines and Binding Components
- Developers of JBI-based integration solutions
What does SE/BC Developers do?

- Develops Service Engines and Binding Components 😊
- 4 interfaces to implement – component management
  - **Bootstrap** - install, uninstall component
  - **ComponentLifeCycle** - start, stop, shutdown component
  - **Component** - query for lifecycle, service description
  - **ServiceUnitManager** (optional) – deploy, undeploy services
- 4 interfaces to use – message exchange
  - **ComponentContext** - component settings, manage service endpoints, manage delivery channel
  - **DeliveryChannel** - send and receive messages
  - **NormalizedMessage** – holds message xml
  - **MessageExchange** – holds the message and state of exchange
- Rest is your creative code – message processing
What does SE/BC Users do?

- Develops integration solutions 😊
- Design and implement Service Units (Business Logic)
- Configure Binding Components (Adapters/Connectors)
- Construct and deploy Composite Applications
- Manage and monitor integration solutions
Agenda

➤ What is Integration
➤ What is JBI
➤ OpenESB
➤ Demonstration
The OpenESB Project

- Open Source Enterprise Service Bus (ESB)
- Java™ Business Integration (JBI) Reference Implementation
- Uses NetBeans 6.x IDE at development time
- Deploys runtime solutions to Glassfish/Sun App Server 9

- http://open-esb.org/
OpenESB
OpenESB Architecture

- Service Engines
- Normalized Message Router
- Binding Components

JBI Core Services
- Installation
- Deployment
- Management

System Management

J2SE™/J2EE™ Platforms
Normalised Message Router (NMR)

- Backbone of JBI
- Routes between JBI Components
- Uses WSDL-based service descriptors
- Service Providers and Consumers are de-coupled
- Messages exchanged in “Normalized” format
- Message Exchanges are short-lived (JBI 1.0)
- Implements WSDL 2.0 Message Exchange Patterns
Normalised Message

- Protocol-specific messages are normalized and routed
- SEs and BCs do not talk directly to each other
- A normalized message consists of 3 parts
  - payload
  - metadata
  - message attachments
- Protocol-supplied message context is preserved
- Non-XML messages can be Encoded/Decoded
Message Exchange Patterns

- **One-Way**: consumer issues a request to provider with no error (fault) path provided.

- **Reliable One-Way**: consumer issues a request to provider. Provider may respond with a fault if it fails to process request.

- **Request-Response**: consumer issues a request to provider, with expectation of a response. Provider may respond with a fault if it fails to process request.

- **Request Optional-Response**: consumer issues a request to a provider, which may result in a response. Consumer and provider both have the option of generating a fault in response to a message received during the interaction.
System Management and Administration

- Java Management eXtensions (JMX) based
- Provides life cycle services for SEs and BCs
  - Install SE/BC
  - Start SE/BC
  - Stop SE/BC
  - Shutdown SE/BC
  - Uninstall SE/BC
- Provides life cycle services for composite applications
  - Deploy Service Assembly
  - Start Stopping Service Assembly
  - Shut down Service Assembly
  - Undeploy Service Assembly
Service Engines and Binding Components

- Service Engines (SEs) are pluggable containers
- Binding Components (BCs) are pluggable components
- SEs and BCs provide and/or consume services
- SEs host Service Units (SUs)
- Service Units (SUs) implement business logic
- BCs provide the ‘knowledge’ or external systems
- BCs provide the ‘knowledge’ of communication protocols
Available Components (March 2008)

- Aspect SE
- BPEL SE
- Data Mashup SE
- Encoding SE
- ETL SE
- IEP SE
- Scripting SE
- SQL SE
- WLM SE
- XSLT SE
- Mural
- CICS BC
- CORBA BC
- DCOM BC
- eMail BC
- File BC
- FTP BC
- HL7 BC
- HTTP BC
- IMS BC
- JDBC BC
- JMS BC
- LDAP BC
- MQ Series BC
- MSMQ BC
- RSS BC
- SAP BC
- SIP BC
- SMTP BC
- SNMP BC
- SWIFT BC
- TCPIP BC
- UDDI BC
- XMPP BC
OpenESB NetBeans Tooling

- SOA Projects category
- JBI Module projects – Service Units for SE
- Composite application project – Service Assembly
- WSDL + Extensions – Service Unit artefact for BC
- SU Editor – Service Unit artefact editor for BPEL, SQL, …
- WSDL Editor – Service Unit artefact editor for BC
- CASA – Composite Application Service Assembly Editor
OpenESB Tooling – BPEL Editor
OpenESB Tooling – BPEL Debugger
OpenESB Tooling - CASA
OpenESB Runtime

- OpenESB is installed as a life cycle module in GlassFish
- GUI admin console deployed in Glassfish's servlet container
- Reuses Glassfish's transaction manager, clustering support etc.
Agenda

- What is Integration
- What is JBI
- OpenESB
- Demonstration
JMS to File with Web Service Invocation

OpenESB
BPEL SE, File BC, JMS BC, Java EE SE
Summary

- JBI is intended to address major integration problems
  - Lack of standardisation and vendor lock-in
- OpenESB is the framework for developing SEs and BCs
- OpenESB is the framework for developing EAI solutions
- OpenESB is the reference implementation for JBI 1.0
- OpenESB is Open Source and community driven
- You can have and use it now … 😊
For More Information

- **JBI**

- **OpenESB Home**
  - [https://open-esb.dev.java.net/](https://open-esb.dev.java.net/)
  - [https://open-jbi-components.dev.java.net](https://open-jbi-components.dev.java.net)

- **GlassFish Home**
  - [https://glassfish.dev.java.net/](https://glassfish.dev.java.net/)

- **JavaEE SDK**

- **Examples and Demos:**
  - [http://enterprise.netbeans.org/](http://enterprise.netbeans.org/)
THANK YOU

Michael Czapski, Advanced Solutions Architect
Michael.Czapski@sun.com