

JBI 2.0

Directions and thoughts

Peter Walker
Sun Microsystems, Inc.
Session 1841

JAZZ00N07

THE INTERNATIONAL CONFERENCE ON JAVA TECHNOLOGY
JUNE 24 - 28, 2007 ZURICH



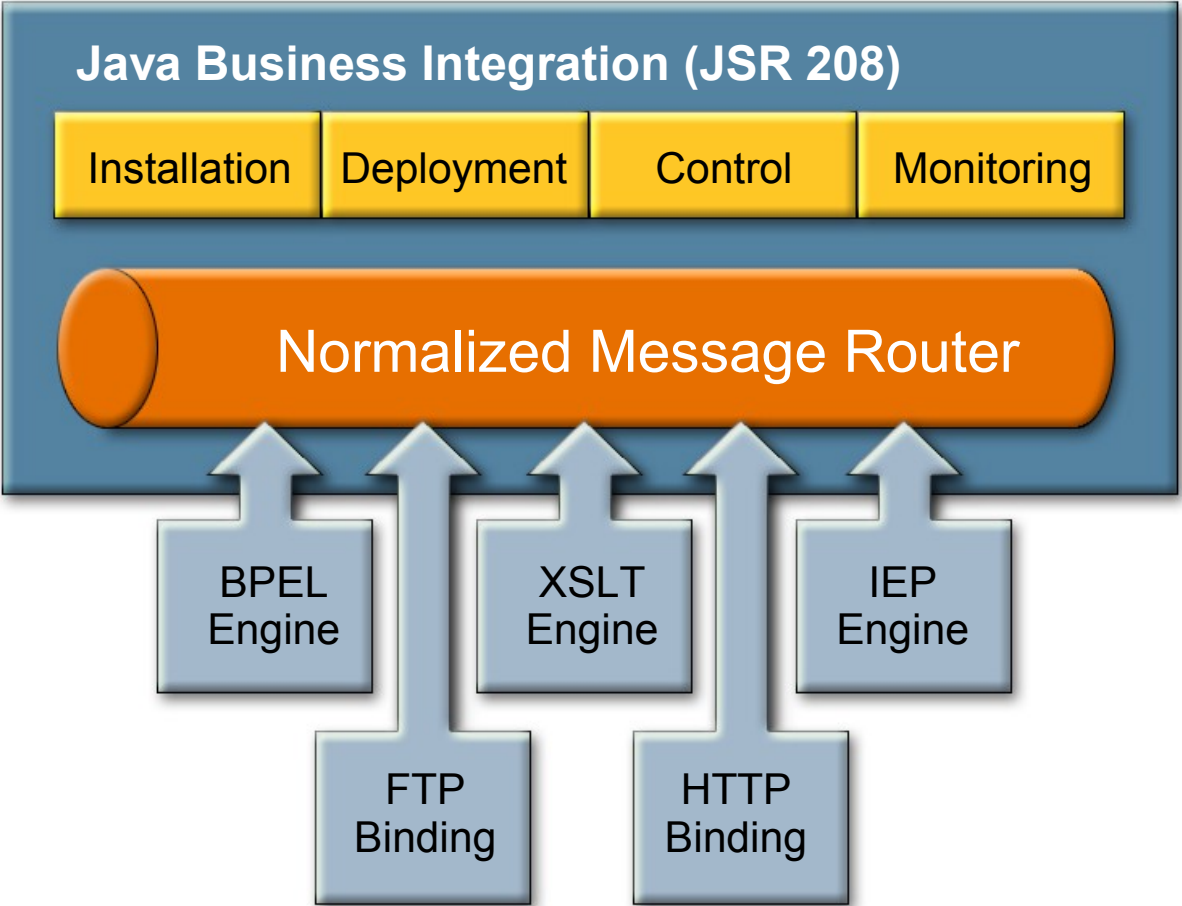
AGENDA

- > JBI 1.0 – very quickly
- > JBI 2.0
 - Status
 - The Expert Group
 - Topics...
- > Q & A

AGENDA

- > JBI 1.0 – very quickly
- > JBI 2.0
 - Status
 - The Expert Group
 - Topics...
- > Q & A

Java Business Integration



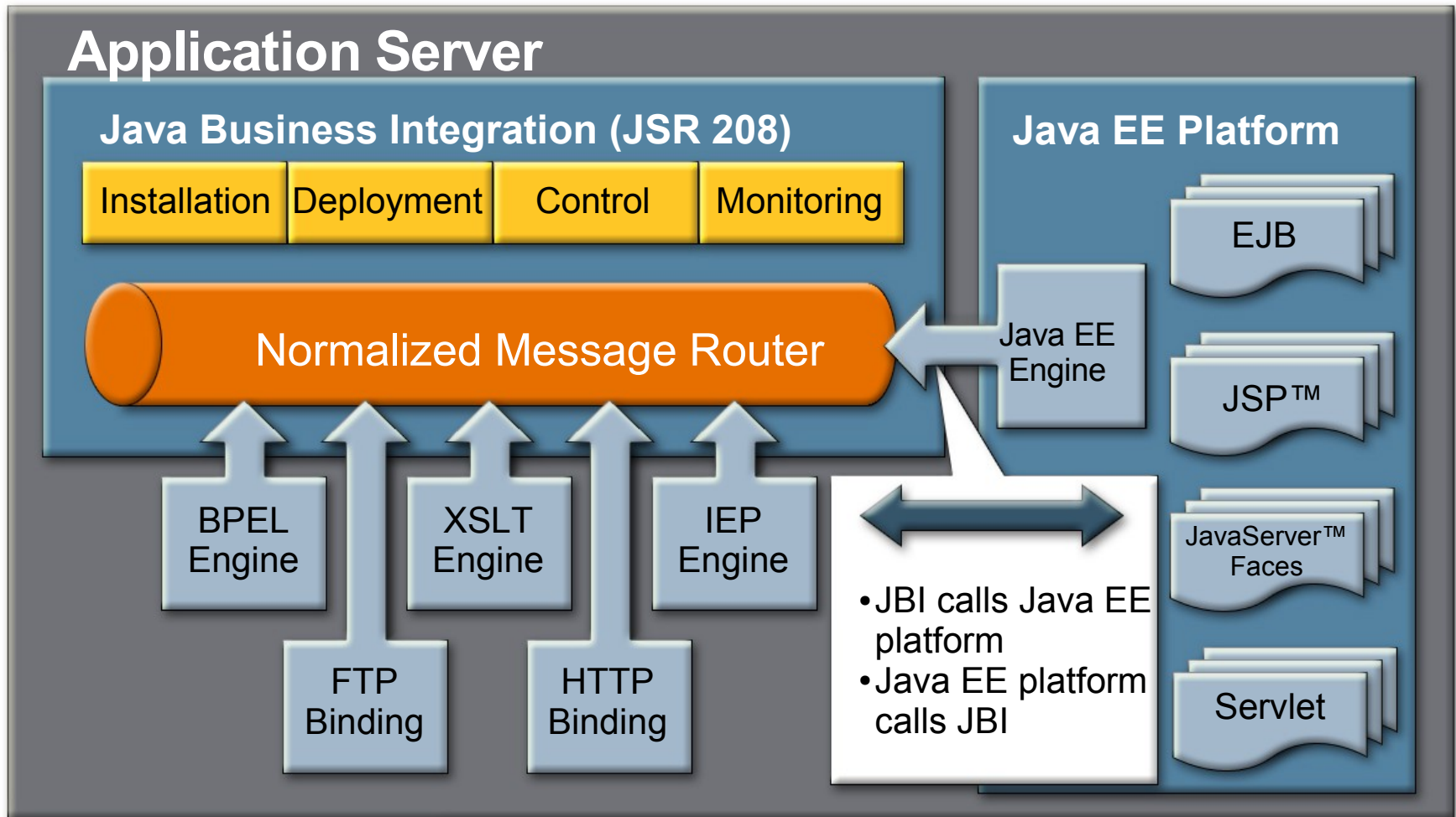
JBI Fundamentals

- > The JBI 1.0 (JSR 208) framework provides the basis for building loosely coupled composite applications by combining services
- > A foundation for SOA integration
 - JBI is inherently asynchronous
 - Components exchange requests and responses through in-memory queues (the normalized message router)
- > Targeted at „code composition“ for SOA Integration

Why Should I Care About JBI?

- > JBI provides a standard to plug in server-side containers for business logic and transformations, e.g.
 - XSLT engine
 - BPEL engine
 - JavaScript™ engine
 - Rules engine.....etc., etc.
- > Choose the best-of-breed containers from the provider(s) you like
- > Because JBI components are interoperable you can chain, mix and match the right technologies to solve your business problems
- > External connectivity (binding components) is separate from the business logic containers (service engines); hence every container has access to all the pluggable external connectivity capabilities

Augmenting the Java EE Platform With JBI



JBI as a Basis for SOA

> Interoperability

- Standard bus (NMR) and message format
- Eliminates “stove-pipe” integration issue between containers
- Mix and match “containers” and “adapters” from different vendors

> Message Based Integration

- Standard message exchange contract based on “abstract” WSDL
- Everything is represented as services, including legacy systems

> Infrastructure for Composite Applications

- Service Assemblies package artifacts for multiple JBI components
- Mapping mechanism between endpoint consumers and providers

Wide Array of Engines and Bindings

The screenshot shows the NetBeans IDE interface. The 'Projects' window on the left displays a tree structure under 'Service Engines' and 'Binding Components'. The 'sun-bpel-engine' is highlighted under 'Service Engines'. The 'Properties' window on the right shows the configuration for the 'sun-bpel-engine'.

sun-bpel-engine - Properties

General	
Description	This is a bpel service ...
Name	sun-bpel-engine
State	Started
Type	service-engine

Identification	
Build Number	200703241026
Spec Version	1.0

Configuration	
DatabaseJNDIName	jdbc/bpelseDB
DebugEnabled	false
DebugPort	3343
EngineExpiryInterval	15
MaxThreadCount	10
PersistenceEnabled	true

sun-bpel-engine
This is a bpel service engine.

Pluggable engines for Eventing, Workflow, Business Processes, Transformation, Java EE platform services...

Pluggable bindings for legacy systems, web services...

A Growing Community



Some of the recent contributions by the open source community in Open ESB and Open JBI Components

SoapUI NetBeans software plugins

TCP/IP BC

XMPP BC

SIP BC

UDDI BC

RSS BC

eMail BC

LDAP BC

Swift BC

CICS BC

CORBA BC

DCOM BC

Scheduler SE (in development)

...and more

Sun is also contributing more

Aspect SE

ETL SE

SNMP BC

...and much, much more!

AGENDA

- > JBI 1.0 – very quickly
- > JBI 2.0
 - Status
 - The Expert Group
 - Topics...
- > Q & A

JBI 2.0 = JSR 312

- > Approved in early April
- > Face to face meeting at JavaOne in San Francisco
- > Regular concalls
- > Intent is to implement reference impl in open source
- > Rough timetable
 - Public Review Q4/2007
 - Final for Java One 2008 ?

Expert Group Members

Adobe

Apache

Borland

Capgemini

DPWN SOP

EBMWebsourcing

IONA

Red Hat Middleware (JBoss)

Pramati Technologies

Sun

TIBCO

Tmax Soft

TongTech

David Hensley

Eric Lu

Brian O'Neill

Eric Smith

Bruce Snyder

Rafaelle Spazzoli

Topics for JBI 2.0

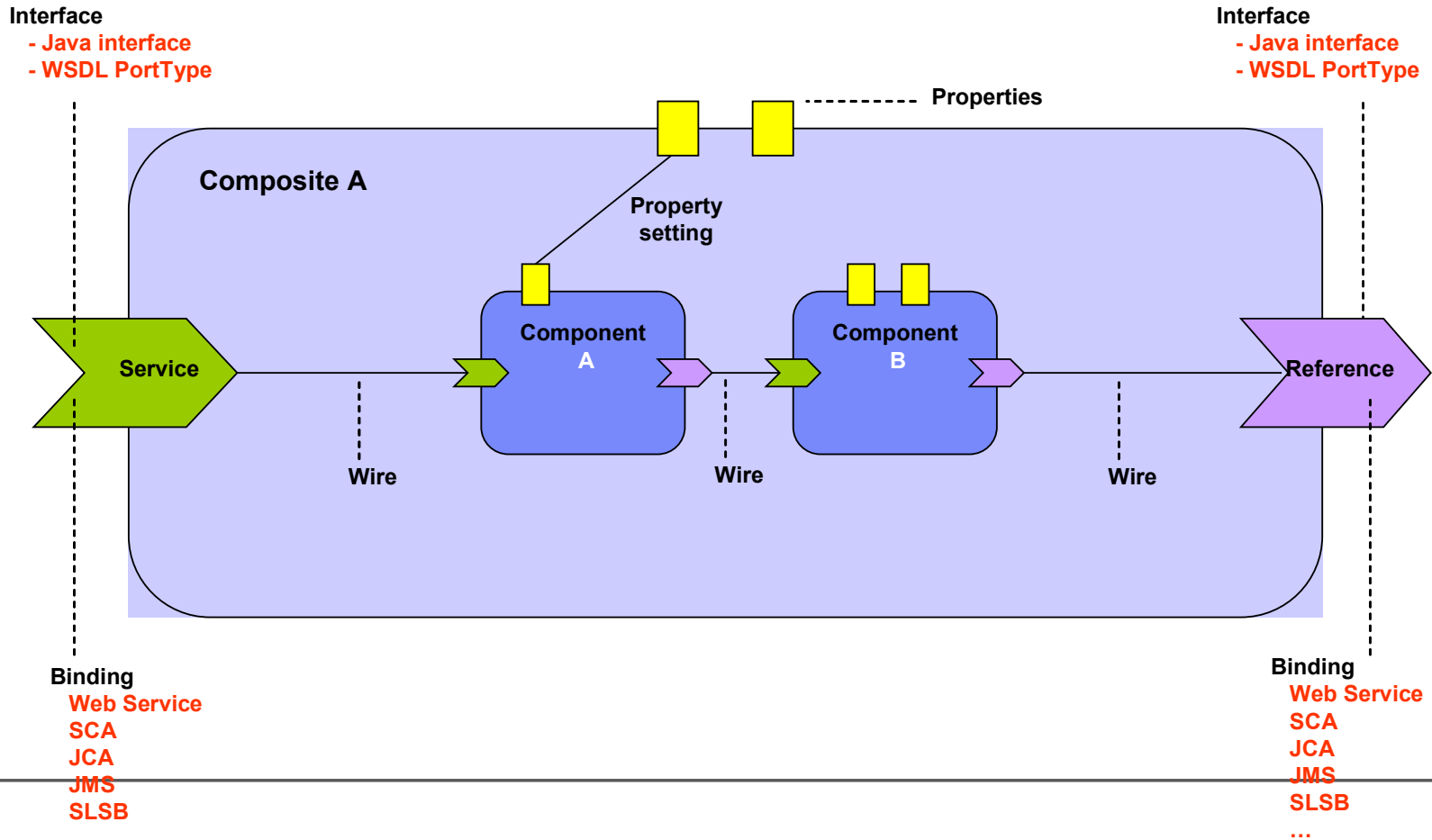
- > JBI and SCA
- > JBI and OSGi
- > New features for JBI

Service Component Architecture (SCA) basics

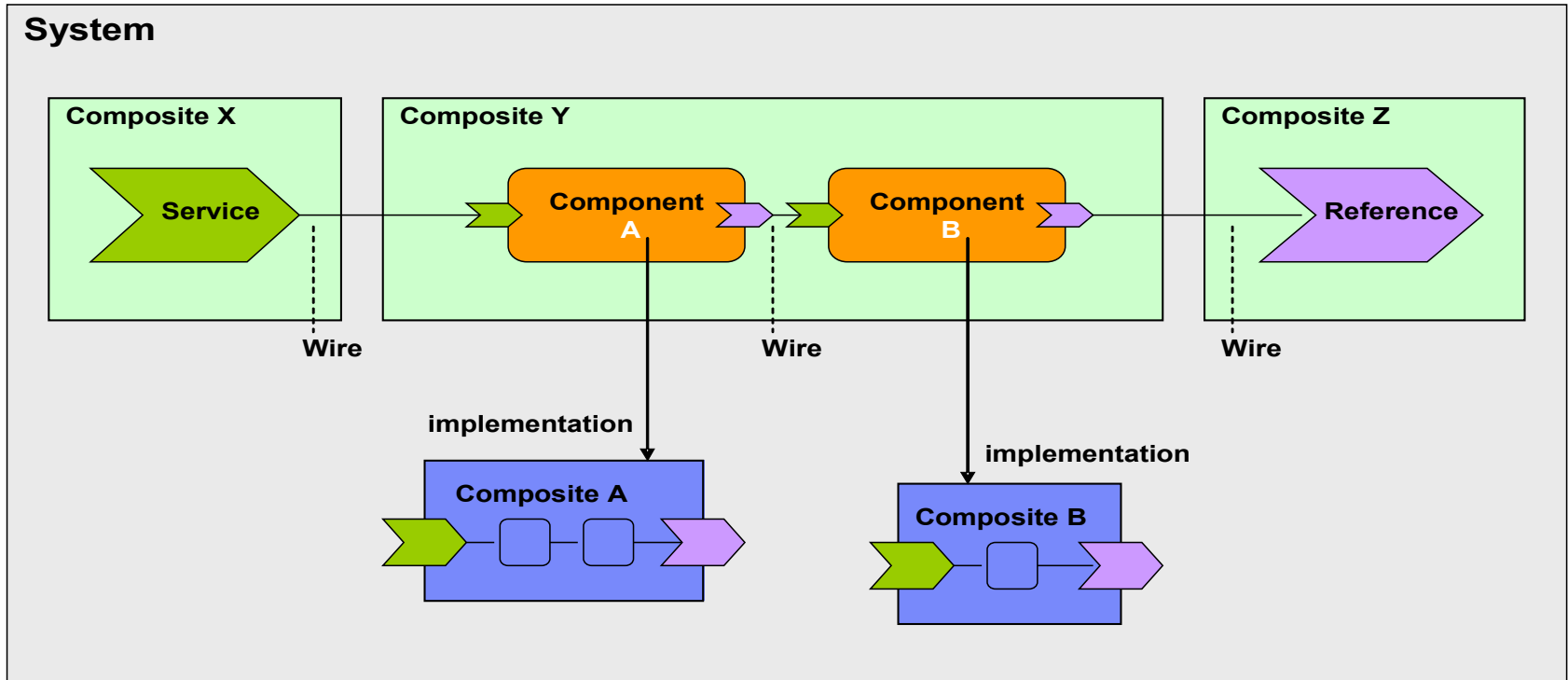
- > New Component Model
 - Composite application creation
- > Multi-language
 - Java, C++, Spring, PHP, Ruby, WSDL; extensible
- > Service Data Objects (SDO)
- > Mainly design-time modelling
 - Some deployment & run-time

New programming model for creating service-oriented components in Java (and C++), and a way to describe how components are assembled into groups called composites.

SCA Composite



SCA System



JB I and SCA

- > JBI can provide key features for a runtime platform for SCA
- > There's very little overlap
- > JBI runtimes should be able to consume SCA metadata
- > Tools should be able to take SCA Composite definitions and create JBI Service Assemblies and Service Units

JBI and OSGi (and SCA again)

- > OSGi defines bundles to provide better modularity
- > OSGi provides a dependency resolution mechanism with version support
- > JBI 1.0 provided limited definitions for shared library handling
- > JBI 1.0 defined Service Assemblies containing Service Units
- > JBI 2.0 will need to include a way for OSGi artifacts to be brought to the runtime (...and SCA artifacts for Composite Applications to be deployed)

New features for JBI 2.0

- > Re-org of spec for multiple audiences
- > Tighter definitions for JBI working with
 - Java EE (transactions, EJB, Servlets etc.)
- > Maintain definition of JBI for Java SE
- > Interceptors
- > Standard interfaces for expected services
- > POJOs as JBI components
- > Runtime Management enhancement
- > Clustering/distribution
- > Fault tolerance and reliability (e.g. Recoverability of messages)

Interceptors: Consensus Items

- > Interceptors provide the ability to affect the processing of a message exchange without
 - Code changes to components
 - Code/configuration changes to applications
- > Interceptors offer an opportunity to address cross-cutting concerns in JBI architecture
- > Administration support is required to register/unregister interceptors in JBI runtime
- > Valid use cases exist

Interceptors: Use Cases

- > Content-based routing
- > Logging/Auditing
- > Validation
- > Security
- > Policy
- > Transaction Injection
- > Caching
- > ... and more

How to Solve this Problem

- > Three approaches have been discussed:
 - Approach 1 : Create a new Interceptor API
 - Approach 2 : Leverage existing component and interaction contracts to provide interceptor functionality
 - Approach 3 : Catch-all which includes pieces of 1 and 2
- > The Expert Group is pursuing a solution based on #2

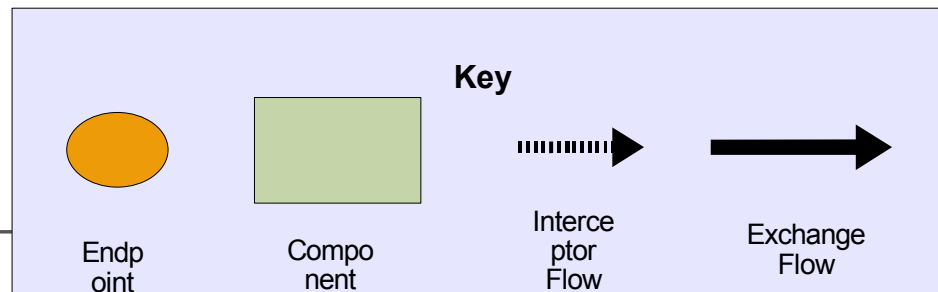
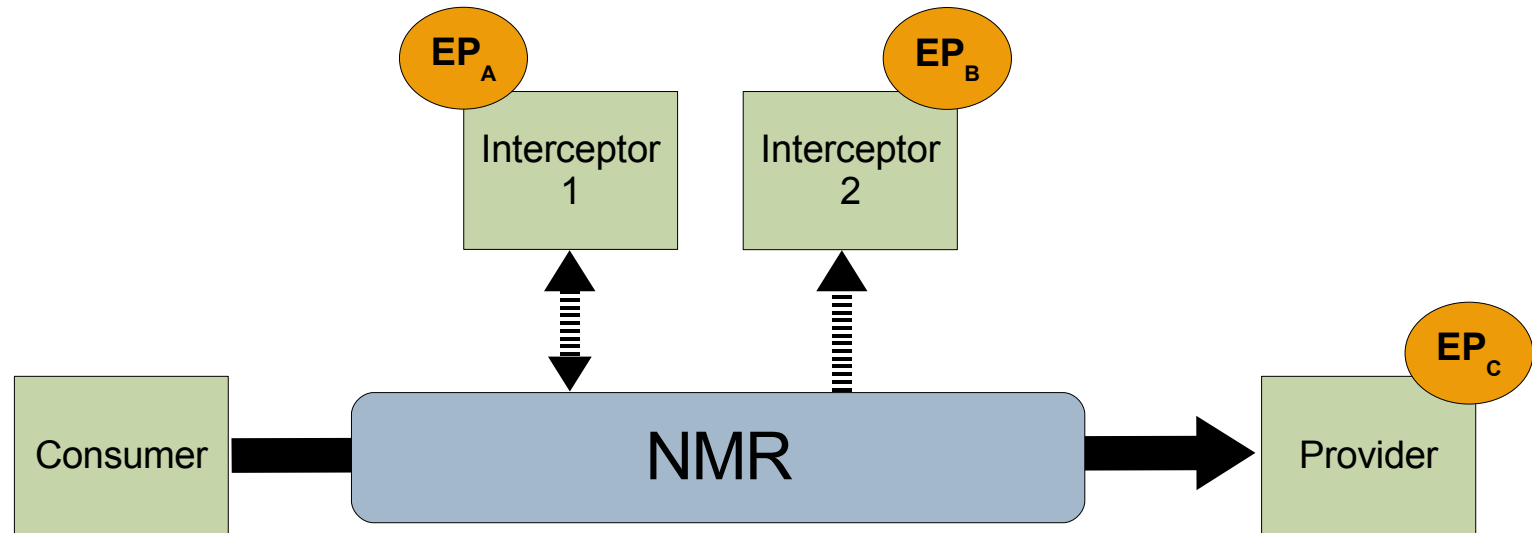
Approach #2

- > Interceptor instances are simply endpoints on the NMR
- > Interceptor implementation is a JBI component
- > Interceptor chains are defined and scoped via standard metadata
 - Registered/unregistered with JBI runtime
- > NMR is the injection point
- > Execution is controlled by the NMR
- > Invocation contract is MessageExchange over a DeliveryChannel

Advantages

- > Reuse of existing contracts for invocation, deployment, and administration
- > Components can function as interceptors without any change
- > Location transparency for interceptor implementation
 - Engines host interceptor logic locally
 - Bindings proxy for external interceptor implementations
- > Interceptor implementation does not have to be Java (gasp!)

Logical View

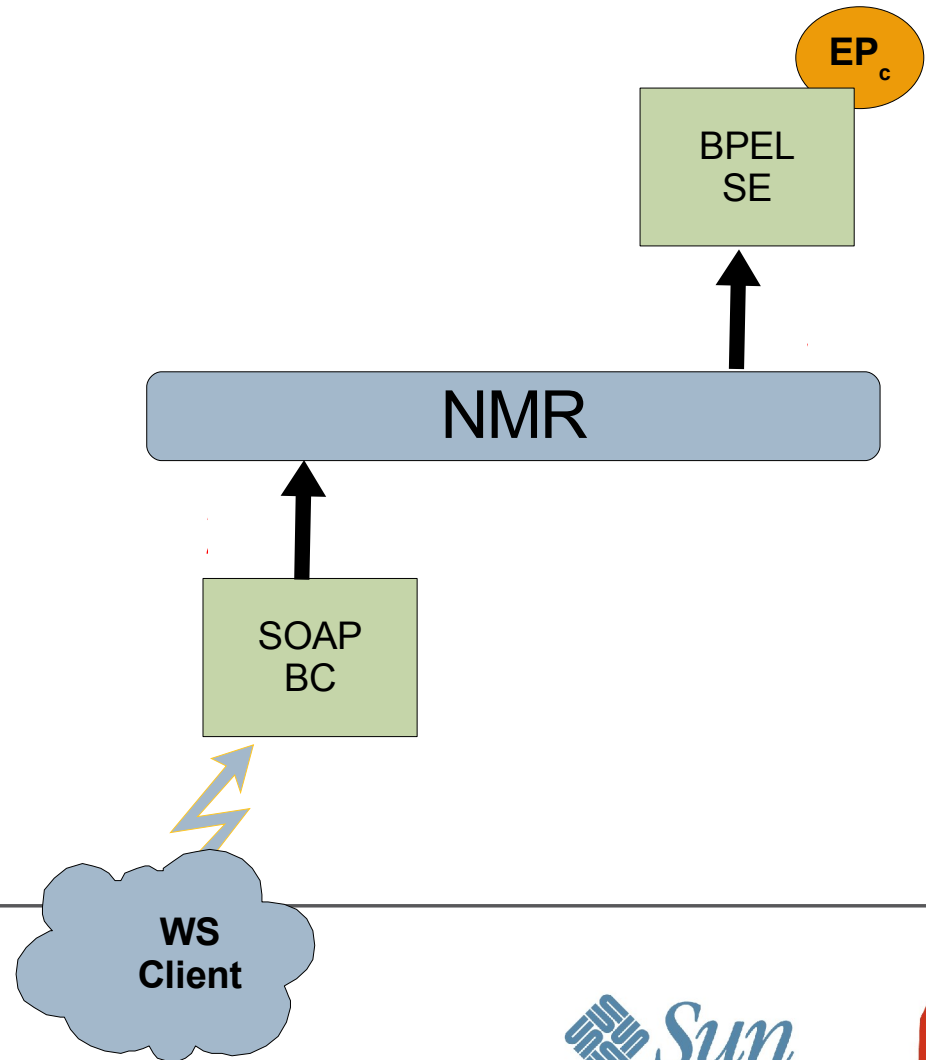


Logical View Discussion

- > Each interceptor activates an endpoint
- > NMR is the injection point, not the Delivery Channel
- > Interceptors execute within the context of the consumer/provider message exchange flow
 - Sequencing and execution managed by NMR
 - Interceptors do not talk to one another
- > Exchange pattern used by interceptor endpoint determines interceptor flow contract
 - InOut provides mutability
 - Fault/Error terminates exchange flow

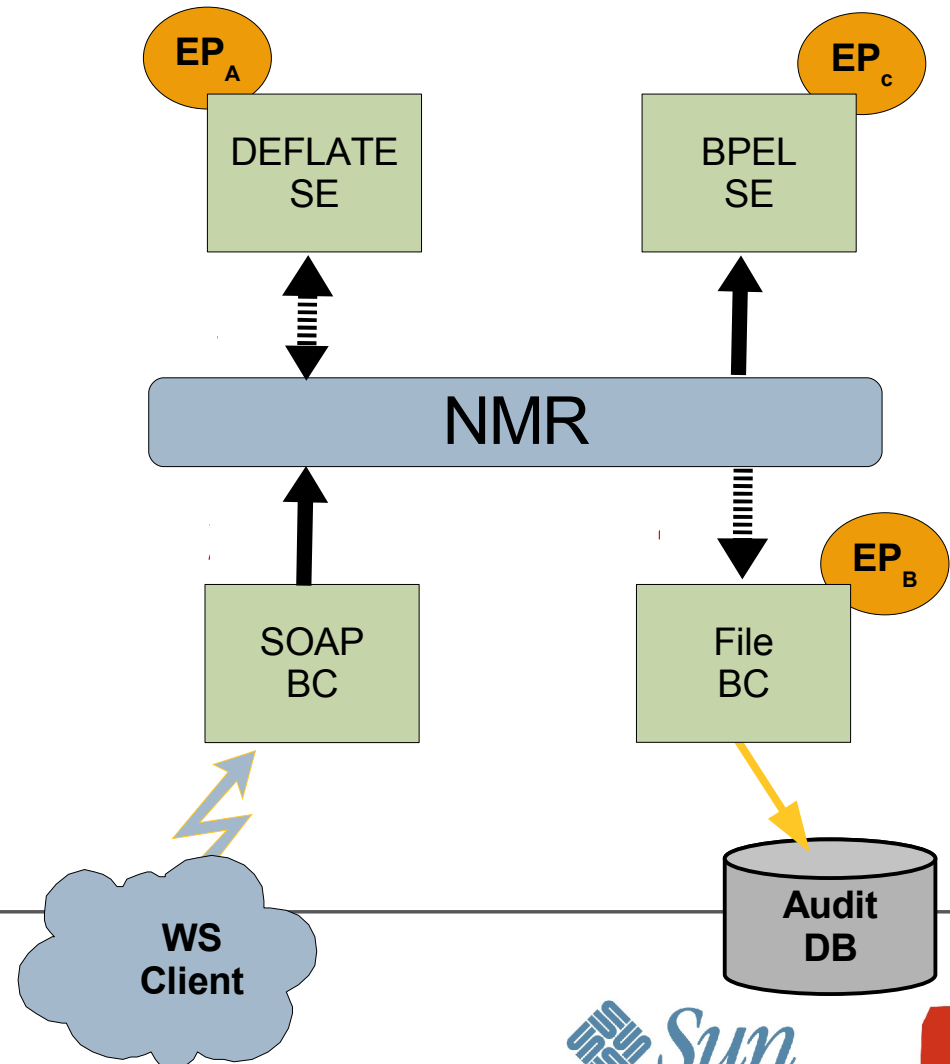
Application View

1. Web service client sends request to SOAP binding component.
2. SOAP BC performs a lookup on provider endpoint EP_c and initiates a message exchange.
3. NMR routes exchange to provider.



Runtime View

1. Web service client sends request to SOAP binding component.
2. SOAP BC performs a lookup on provider endpoint EP_C and initiates a message exchange.
3. NMR invokes first interceptor, which decompresses the incoming payload.
4. NMR invokes the second interceptor, which logs the decompressed payload to a file-based audit DB.
5. NMR routes exchange to provider.



Runtime View Discussion

- > Decompression Interceptor (step 3)
 - Message is changed, so pattern is InOut
 - Interceptor implementation is local
- > Auditing Interceptor (step 4)
 - Message is not changed and failure is not fatal, so pattern is InOnly
 - Interceptor implementation (i.e. audit function) is remote
- > Components can play multiple roles
 - SOAP BC could proxy for external client **and** remote interceptor implementation

Example: Interceptor Instance

```
<interceptor-chain>
  <interceptor service-name="foo:DeflateService"
endpoint-name="epA"/>
  <interceptor service-name="foo:AuditService"
endpoint-name="epB"/>
  <scope>
    <component-scope>
      <component role="provider">
        <identification>bpel-se</identification>
      </component>
    </component-scope>
    <endpoint-scope>
      <endpoint service-name="foo:BpelService"
endpoint-name="epC"/>
    </endpoint-scope>
    <message-scope>
      <message name="IN"/>
    </message-scope>
  </interceptor-chain>
```

Interceptors: Specification Impact

- > Standardize metadata related to interceptor declaration
- > API to register/unregister interceptor declaration
- > NMR rules for interceptor execution
 - Scope resolution
 - Interceptor chaining
 - Fault/Error processing
 - MEP invocation contracts

New features for JBI 2.0 (more subject to time limit)

- > Standardized JBI container
- > Less WS dependency – WSDL optional ?
- > Service versioning
- > Other policy related features ?
- > Should everything be normalized?
- > Better „hooks“ for tooling ?

...and to validate our decisions

- > Support for BPEL was used in JBI 1.0 as a check
- > For JBI 2.0...
 - Composite Applications
 - WS-CDL ?
 - Web 2.0
 - SCA
 - .NET interop
- > ... whilst maintaining the ability of implementers to provide performance optimizations

AGENDA

- > JBI 1.0 – very quickly
- > JBI 2.0
 - Status
 - The Expert Group
 - Topics...
- > Q & A

Peter Walker

Sun Microsystems, Inc.

peter.walker@sun.com

JAZOON07

THE INTERNATIONAL CONFERENCE ON JAVA TECHNOLOGY
JUNE 24 - 28, 2007 ZURICH

