

# Enterprise Service Buses

Where are they going?

*An Oracle White Paper*

*October 2006*

## INTRODUCTION

Historically it has been difficult for developers to find a single tool to develop composite, service-oriented, applications across organizational boundaries. Oracle SOA Suite and Oracle Enterprise Service Bus' (ESB) mission statement is to provide a toolset that leverages adopted standards across the entire suite of integration components including the latest member of the family, Oracle Enterprise Service Bus (ESB).

Relevant ESB wire, definition, and protocol standards include: J2CA, JMS, SOAP, HTTP, WSDL, JDBC, XML/XSD, XSLT, XPATH, and several WS-\* protocols, including WS-Addressing, WS-Security and WS-Reliable Messaging. More recent standards that have evolved relating to service composition include BPEL, Java Business Integration (JBI) and Service Component Architecture (SCA), which enable the assembly and wiring of services independent of their implementation.

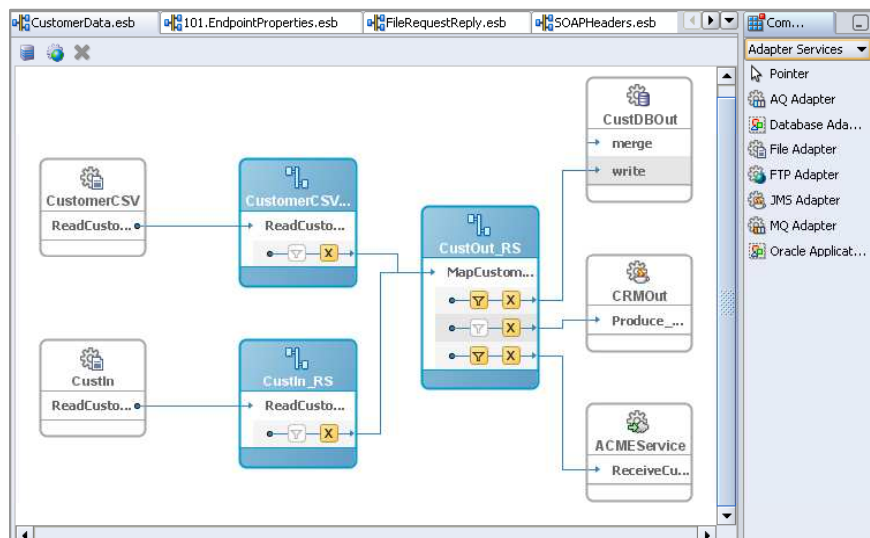


Figure 1: ESB Composite Application designer

A modern ESB should virtualize transport to the service consumer (SC) and guide development with a set of design patterns and best practices to provide the necessary infrastructure to support other components. ESB's can no longer focus exclusively on Web Services or message driven (JMS) use cases. They must also provide the ability to "service enable" non-XML based endpoints such as

mainframes, flat files, ETL procedures and enterprise applications such as SAP, PeopleSoft, Oracle or Siebel all within a single tool. To be productive in any business climate, developers must be able to design, deploy and monitor full end-to-end enterprise composite applications without switching between different consoles or tools.

Oracle is committed, through direct involvement with the respective Standard bodies, such as JCP and OASIS, to foster these standards and provide a comprehensive SOA infrastructure, consisting of best of breed components that will push your organization to react faster to business changes and leverage your organizations existing and future technology investments.

## **EVOLVING REQUIREMENTS**

Many of the standards within a Service Oriented Architecture (SOA) are not new, yet countless organizations continue to struggle with leveraging them to meet their needs. Is it because they are not quick to adapt or retrain their workforce? Possibly, but more likely it is the opposite; they have jumped head first into a new technology or standard without charting the best way to get there or worse, they haven't defined goals for the exercise. One way to avoid this pain is be to partner with a vendor who promotes standards and interoperability with 3<sup>rd</sup> party products that will benefit you by avoiding vendor lock-in.

Modern applications need to be built for change such that if an external endpoint changes say because your organization is adopting a new federal regulation, it does not disrupt business and add to the backlog of already unfinished IT projects. Using an ESB mediator service to shield or de-couple a business process from a database transaction delivers flexibility to an organization faced with upgrading a backend database. Mediators provide XSLT transformations with XPATH routing rules to designate the format and location for a document to be routed. Additionally, a mediator should have "smart bindings" that uses the fastest path to route the document to the service endpoint. An example would be to use Java bindings on a local system and RMI or HTTP for remote endpoints.

The Java Message Service (JMS) specification defines interfaces for point to point and publish/subscribe patterns that have the advantage of naturally de-coupling the publisher from one or many subscribers. JMS servers are fast, provide guaranteed delivery, are transactional and provide multiple options for persisting messages: memory, file system or database. JMS is the protocol of choice for asynchronous publish/subscribe messaging, but for synchronous request/reply interactions SOAP/HTTP is typically a better fit. A modern ESB therefore should provide a lightweight abstraction on top of both JMS and SOAP style web services.

The J2EE Connector Architecture (J2CA) standard enables Java applications to use "Resource Adapters" to connect to external systems in a unified way. Service enabling non-XML interfaces and legacy applications is made possible by

connecting on the provider side through native formats and protocols. In addition, J2CA makes it easy for vendors to provide Developer Kits (SDK's) to write custom adapters for connecting to non-standard, custom or proprietary interfaces while staying within the specification and thus providing a standardized interface to the service consumer (SC).

The Web Service Definition Language (WSDL) defines interfaces for invoking services described with XML schemas and protocol bindings of the endpoints. More than any other specification, it is WSDL, which initially drove the adoption of SOA through its ability to loosely couple an interface to its actual implementation. All ESB services must have a WSDL definition to be invoked in a standard way. For performance reasons, they should offer multiple protocol bindings such as HTTP or Java, which would be selected, based on the environment.

Services must participate in global transactions when they are called from Java, WSIF or a JCA adapter. In this case the message flow behaves as an internal Java pipeline where all endpoints must succeed, or will be rolled back in an XA manner. Each service should provide the ability to configure transaction rollback behavior.

All of these components: 1) J2CA Adapter endpoints 2) mediators or routing services and 3) standard-based web services should share a common development and monitoring environment. Developer productivity decreases when users are forced to switch back and forth between multiple design/debugging environments to accomplish a single task.

## **STANDARDS AND PROPRIETARY SOLUTIONS**

Unfortunately, the ESB world is divided when it comes to adopting an overarching standard to define how services should be brought together in a distributed environment. Depending on the use case and depth of integration required, several approaches could be used to achieve a loosely coupled SOA-based integration. The following is a list of possible ESB architectures.

- Proprietary
- Business Process Execution Language (WS-BPEL)
- Java Business Integration, JBI (JSR-208)
- Service Component Architecture (SCA)

### **Proprietary Approaches**

While established standards can help an organization build vendor independent integration solutions, ESB functionality is still heavily based on proprietary custom or in-house solutions. Though expensive to maintain and laborious to build, they often are still the workhorse when it comes to brute force methods. Over time these solutions will be replaced by standards based products but perhaps not until one of the following emerges as a universally adopted standard.

## WS-BPEL Specification

[..] Business processes can be described in two ways. Executable business processes model actual behavior of a participant in a business interaction. Business protocols, in contrast, use process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior. The process descriptions for business protocols are called abstract processes. [..]

BPEL4WS is meant to be used to model the behavior of both executable and abstract processes. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces.”

## Business Process Execution Language (WS-BPEL)

In a service-oriented world, services are not used as stand alone applications. Usually they are composed into bigger processes to solve a certain problem, like the processing of a multi-step loan application. With this in mind the WS-BPEL specification was created as a common standard for how processes should be assembled based on services, and defined through WSDL. An orchestrated BPEL process can contain a single step, just invoking a service, or hundreds of these steps, called activities, to accomplish complex transformation, sophisticated join conditions, as well as loops and asynchronously.

Oracle BPEL Process Manager combines a complete implementation of the WS-BPEL 1.1 specification, as well as many needed features to allow complex processes to take place. One example is a highly configurable manual task workflow that fits into the WS-BPEL service orientation. Not every service that exposes a WSDL as its interface implies a SOAP/HTTP binding. By using the Apache Web Service Invocation Framework (WSIF), Oracle BPEL PM and Oracle ESB can be extended and offer several out of the box bindings for common cases such as J2CA, SOAP, EJB, and plain Java. Oracle BPEL PM's communications with Oracle ESB are transactional. Using WSIF, they can run within the same global transaction, allowing for tight integration of mission critical applications.

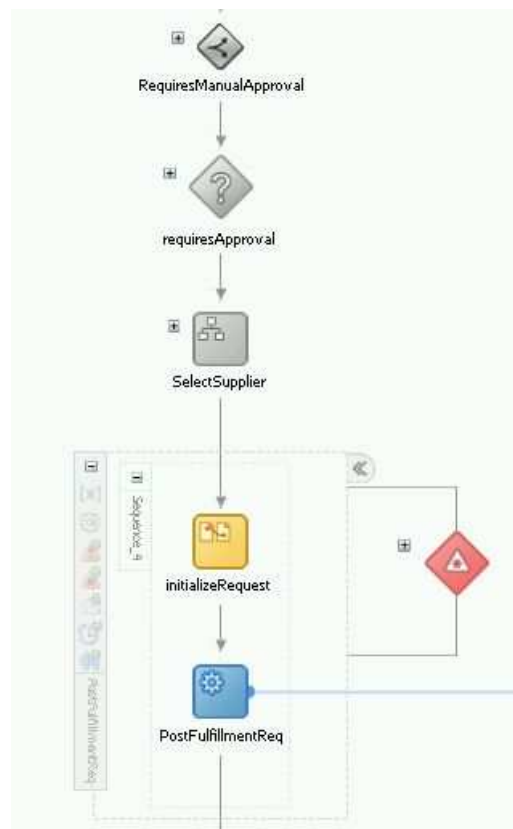


Figure 2: BPEL process within Oracle BPEL Process modeler

## JSR-208 JBI Specification

*“Java™ Business Integration (JBI) seeks to address this problem by creating a standards-based architecture for integration solutions. This infrastructure allows third-party components to be “plugged in” to a standard infrastructure, and allows those components to interoperate in a predictable, reliable fashion despite being produced by separate vendors. It is anticipated that this ability to interoperate will create a multi-vendor “ecosystem” which will give rise to large pool of integration-related technologies that can be sourced by end users.”*

## Java Business Integration

JBI was introduced in 2004 to solve the problem of assembling bits and pieces of an application in a standardized manner and assure loose coupling between bindings and service engines. The power of JBI is to guarantee certain behavior once the assembled system is executed but at the same time provide an extendable infrastructure that can grow with the needs of the organization.

Looking deeper into the standard, JBI provides a runtime specification on how and by what means systems are connected and transparently talk to the outside world through bindings called Binding Components. A normalized message containing properties, payload and attachments travels between the system components and is independent of any transport technology. Once it reaches the Binding Component it is transformed to its agnostic counterpart, which could be a JMS message, a file, SOAP or other format. Service engines enable functionality such as transformations, BPEL process execution and intelligent routing. Operations are defined through WSDL in Service Units (SU). The central part of a JBI compliant container is a router, called Normalized Message Router (NMR), which ensures that messages travel the correct route based on definitions called service assemblies that contain wires between SU's.

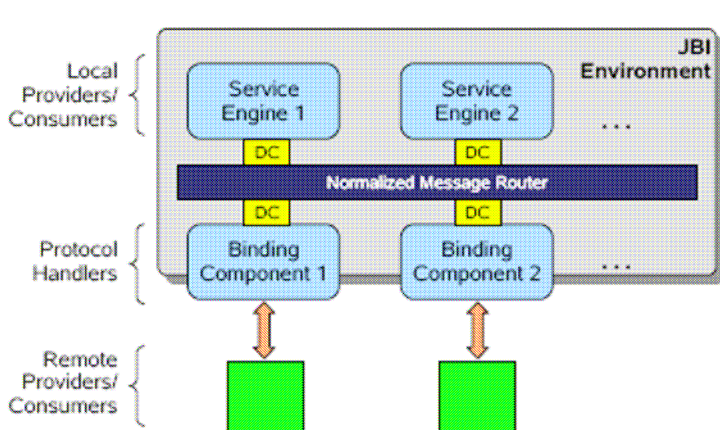


Figure 3: JBI runtime architecture

Oracle provides a JBI implementation, called Oracle Containers for Java Business Integration, which can be used to deploy, debug and integrate JBI based engines and assemblies into bigger applications built on top of BPEL PM and ESB.

## Service Component Architecture

WSDL has been a great step forward in terms of increasing connectivity and interoperability between applications. However, WSDL only focuses the interface of a service, and does not capture external service dependencies or policy configurations.

### SCA Draft Specification

*“Service Component Architecture (SCA) is a set of specifications which describe a model for building applications and systems using a Service-Oriented Architecture. SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services.”*

*“Service Component Architecture (SCA) provides a programming model for building applications and systems based on a Service Oriented Architecture. It is based on the idea that business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need.”*

Service Component Architecture (SCA) provides a programming model for building applications and systems based on SOA. It is based on the idea that business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need. SCA provides a model both for the creation and composition of services including the reuse of existing application function within SCA compositions. SCA goes a step beyond WSDL by defining both a service component and assembly model. A service component allows service developers to define not only the interface to a service but also the dependencies of a service on other services plus the policies associated with those interactions such as transaction, security, reliable delivery etc. and potential configuration knobs that a service might expose.

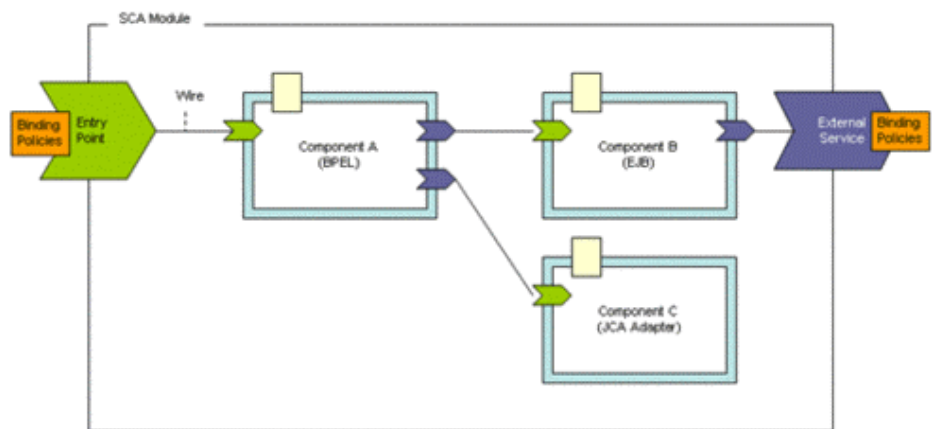


Figure 4: SCA composite, with different components

SCA does not interfere with how a service component is implemented. You can implement a service component in Java, BPEL, a rule language, etc. The SCA module allows a developer to assemble a set of service components, resolve reference dependencies and apply policies. This is a great step forward because current SOA platforms force developers to capture these references in either proprietary deployment descriptors or even worse hard code them into the service implementation itself.

SCA defines a framework that makes it easier for developers to implement Java service components including: annotation for converting a POJO into a service and annotation for asynchronous/session management. Oracle’s next generation SOA infrastructure is based on SCA and will offer sophisticated means to

compose applications, processes and assemblies, into transparent and easy to monitor solutions.

## **ORACLE'S PRODUCT STRATEGY**

Choosing standards is important for the success of your organization. The trick is to map your business needs to technical requirements so that when an applicable standard evolves, you can proceed with adoption. Perhaps the most critical task is recognizing when a standard is mature and having the agility to exploit it. This is where vendors such as Oracle can help you. Oracle invests heavily by leading and participating in literally thousands of standards involving all of the leading standards organizations. But participation alone is not sufficient, many standards die on the vine and only when a standard is truly ripe is it adopted by customers and vendors. Oracle's goals for standards adoption are exactly the same as our customers, which is to utilize standards to better integrate applications and products.

Oracle has been a driving force in the JBI specification from the beginning. It offers clear advantages to the previous generation of proprietary integration products so much so that Oracle initiated projects with strategic customers to implement the spec. That work result in our embeddable JBI implementation that is in production today at a major Telco customer. Oracle is a major contributor with other leading vendors in the SCA assembly model. SCA is not tied alone to Java as foundation language and moving forward, customers will be able to combine the three mentioned standards, embedded into a single platform, to build their next generation integrated SOA applications the same way Oracle is leveraging Fusion Middleware for Fusion Applications. Customers win by letting Oracle be their proxy for standards adoption.

## **CONCLUSION**

Customers implementing SOA are learning the business realities of a distributed enterprise. Decoupling application interfaces is a good first step but standards based tools for developing and monitoring distributed composite applications are a requirement for a well-integrated SOA suite. A comprehensive best of breed SOA suite must go beyond implementing SOAP or JMS to get various components to communicate. They must provide a single standards-based development tool that is the common user interface for all development tasks. Additionally, application-monitoring consoles must support common security recruitments such as single sign-on and centralized policy management. Only then can organizations realize the value of SOA by integrating their business operations with their technology platform.

Technology standards are a proven, effective way to overcome vendor lock in to costly proprietary products and solutions. They virtually guarantee organizations the flexibility they need to run an efficient business but only if management has clear goals and a path to selecting the standards that are highest value and lowest

cost. Choosing the wrong standard can be just as expensive as sticking to outdated proprietary applications. This is where Oracle's strategy can protect you from the tedious standards selection process and virtually guarantee you success in leveraging only high value adopted standards for your enterprise

### **MORE INFORMATION**

For more information on Oracle's SOA Suite visit <http://www.oracle.com/soa>

- WS-BPEL specification group

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsbpel](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel)

- Open SOA Initiative

<http://www.osoa.org/display/Main/SCA+Resources>

- Java Business Integration (JBI)

<http://www.jcp.org/aboutJava/communityprocess/final/jsr208/index.html>

# ORACLE

Oracle Enterprise Service Bus  
October 2006  
Author: Dave Berry, Clemens Utschig  
Contributing Authors: Demed L'Her

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
[oracle.com](http://oracle.com)

Copyright © 2006, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.