Work sample from:

William Blank

Technical Writer and Communicator with Business Analyst, Web Interface Skills

VISA Prototype Enterprise Architecture

Integration Architecture

Document Id: VEA50-01

Version: 1.0

Date: 1-May-07

Versioning, History

Version	Contributors	Comments
1.0	Bill Blank	Complete draft, edit
0.1	<xxx xxx>	Initial Draft

Tables of Contents

Table of Figures

1 Overview

The Integration Architecture is based on the principles of JBI, J2EE, JCA standards and JCAPS product suite. The integration domain provides:

- Connectivity to the resources (adapters)
- Guaranteed message delivery (JMS)
- Business process orchestration (BPEL) with human interaction and monitoring
- B2B protocol framework and the container for transformations.

The objective of the Integration Architecture is to identify re-usable integration design patterns and then provide a reference implementation for these patterns. The advantages of integration patterns are:

- Standardized integration process.
- Shortened development cycles by providing reusable code and architecture.

2 Purpose and Audience

The purpose of this document is to explain the integration architecture using JCAPS, JBI, J2EE, JCA and SOA principles. The target audience is middleware architects and developers with SOA, J2EE, JCA, and A2A and B2B integration background.

3 Scope

The scope of the Integration Architecture is to provide guide lines on how to use the frameworks and patterns for integrating the applications and businesses using JCAPS product suite and SOA. The following are not covered:

- Web services security architecture
- Message sequencing
- Reporting
- Generic error handling. Error handling varies from one application's requirements to another.

4 References

1. JSR 208: JavaTM Business Integration (JBI) (http://www.jcp.org/en/jsr/detail?id=208):
2. JCAPS User Guide(eGate_User_Guide.pdf)
3. JCAPS Deployment Guide(Deployment_Guide.pdf)
4. JCAPS Naming Standards document

5. Web Services Invocation Framework(http://ws.apache.org/wsif/)
6. J2EE and JCA specifications
7. SUN ICoE EBI Patterns List v3.doc

5 Architecture Considerations

The following are considered in order to create this architecture:

- JCAPS SOA features
- JBI specification
- J2EE specification, JCA and patterns
- UDDI
- Web services security
- Web services invocation framework

We assume that this architecture will be adapted to meet the future integration needs.

6 Integration Architecture

The following diagram shows the integration architecture logical view (Figure 1):

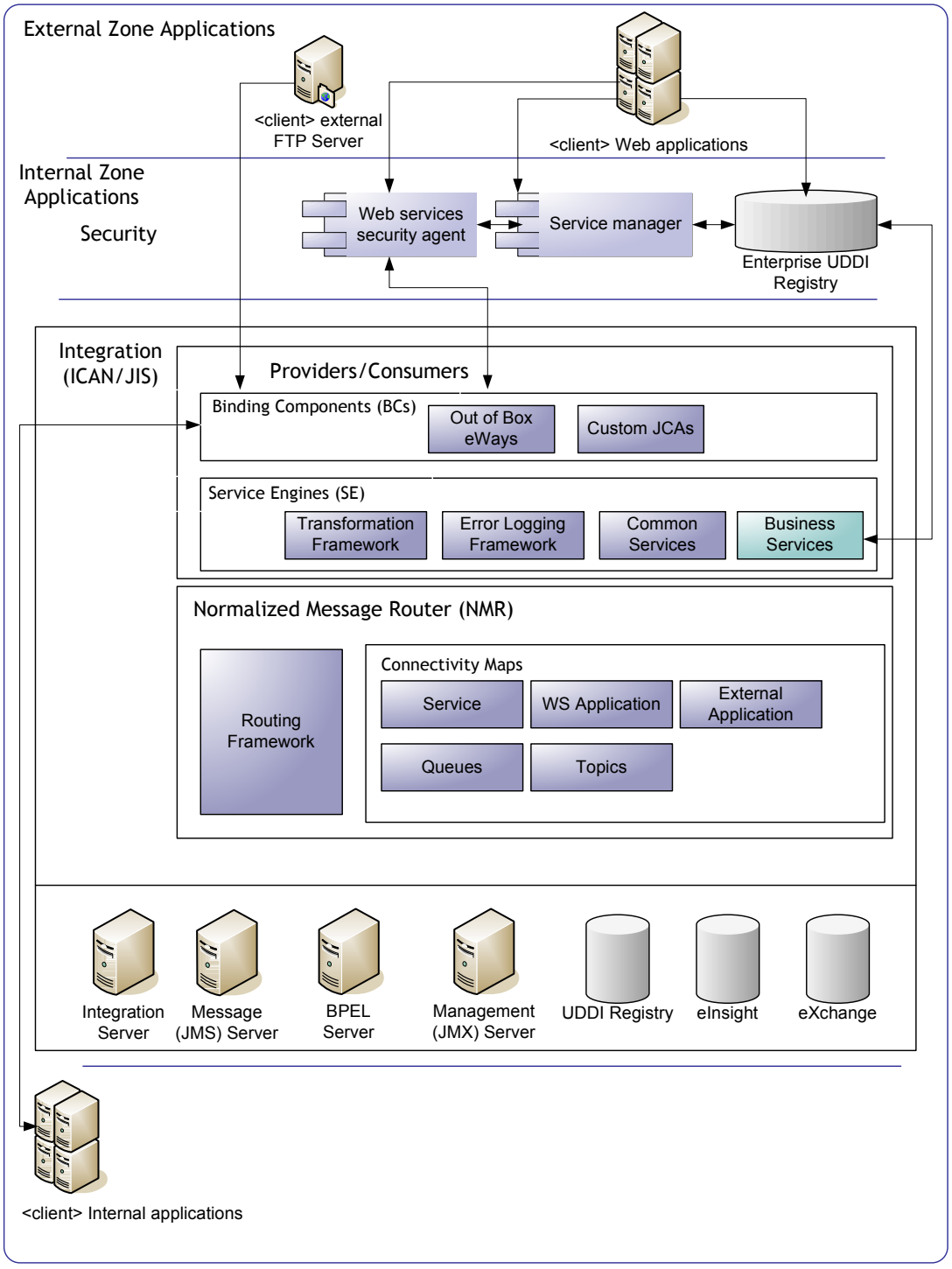

Figure 1: Integration Architecture Logical View – (VEAA0-01)

The enterprise service providers publish their WSDLs to the enterprise UDDI registry. As shown in the diagram, the external zone client applications uses the UDDI registry for discovering available web services. The web service requests are forced to pass through the security layer. The web services security are provided by the service manager and web services security agents. The A2A and B2B integration are implemented using JCAPS product suite and custom libraries developed on industry standards such as JBI, J2EE and JCA. The JMS is used for exchanging messages in a

persistent manner for guaranteeing the message delivery. The web service clients can use WSIF to avoid code changes every time the binding protocol changes.

The goal of the architecture is to provide integration architecture using JCAPS and SOA concepts that interoperate through the method of message exchange. The message exchange model is based on the Web Services Description Language (WSDL). JCAPS product suite gives the ability to implement functions that act as intermediaries to route messages from one component to another. This separation of the participants in a message exchange is very important in decoupling service providers from consumers. This decoupling is highly desirable in service oriented architectures and in integration solutions in particular. In this WSDL-based, service oriented model, the components are responsible for providing and consuming services. By providing a service, a component is making available one or more functions that can be consumed by other components. Such functions are modeled as WSDL operations, which involve the exchange of one or more messages.

The Enterprise UDDI registry is used to publish the WSDLs. The service consumers query the UDDI registry to discover the available services and invoke them using WSIF. With the JCAPS out of box features, it is not possible to dynamically query the UDDI registry and invoke the web service; however, it can be made possible by developing custom Java code using WSIF. The JCAPS default mechanism to invoke or create a web service is by creating the WSDL OTD from the WSDL and using the web services application in the connectivity map.

The providers of services are not responsible for service authentication and authorization. The web services security agent that is in front of the service provider validates the security authentication and authorization tokens in the header of the SOAP message. The security agents are a part of the third party security product. The service providers are deployed in such a way that all the requests are passed through the security agent. This functionality can be achieved by configuring the firewall with appropriate rules. The authentication information required at the backend application (external provider) is included in the body of the message and the service implementation passes it to the backend.

If a service should be exposed external to the enterprise, then a web services using SOAP over HTTP is preferred. If the services are implemented as an integral part of the component, then JMS will be sufficient to pass messages to the service. The asynchronous web service can be implemented by receiving a SOAP message over HTTP and then sending it to a JMS component. The SOAP messages can be persisted in JMS queue inside the web service client before they are sent; in this way they are persisted until the HTTP message exchange is successful.

The portions of the diagram listed below require custom development.

- The reusable frameworks such as transformation, error logging and routing
- Common services such as logging, auditing and notification
- Custom JCA connectors

The business services shown in teal color are application-specific services and are developed using frameworks, common services and custom JCA connectors.