

Sun Open Suite for SWIFT Solution

Alliance Integrator Extension Program

May 2009

Table of Contents

About Alliance Integrator.....	3
The Sun Open Suite for SWIFT Solution.....	3
Combining Forces.....	3
Sun’s Alliance Integrator Extension Program	4
The Alliance Integrator Extension Software Package.....	5
The Alliance Integrator Extension License	6
The Alliance Integrator Extension Training	7
More Information	7
Appendix A: The Alliance Integrator Extension Training	8
Appendix B: Extension Software Package Detailed Description	9

About Alliance Integrator

Integrator is a SWIFT-specific integration layer designed to help firms integrate business applications with SWIFT. Licensed as an add-on to Alliance Access, this application integration framework is designed and built for SWIFT users by SWIFT, and is sold, supported, and maintained by SWIFT as well.

Integrator has been built using Sun's Java™ CAPS Financial EAI software, which includes GlassFish™ Enterprise Service Bus (ESB).

For more information about Alliance Integrator, please contact SWIFT.

The Sun Open Suite for SWIFT Solution

The Sun Open Suite for SWIFT solution demonstrates Sun's experience in the SWIFT area, and comprises the following products, all of which have been optimised to work together with the SWIFT network and the Alliance product line.

- Sun Java CAPS, with GlassFish ESB
- Sun Identity Management Suite
- Sun Solaris™
- Sun Solaris Clustering
- Sun SPARC® and Intel systems

The Open Suite for SWIFT solution provides companies with a complete SWIFTNet access infrastructure and just one contact for support, lowering project risk and cost.

For more information about the Sun Open Suite for SWIFT solution, please contact Sun Microsystems at SWIFT@sun.com.

Combining Forces

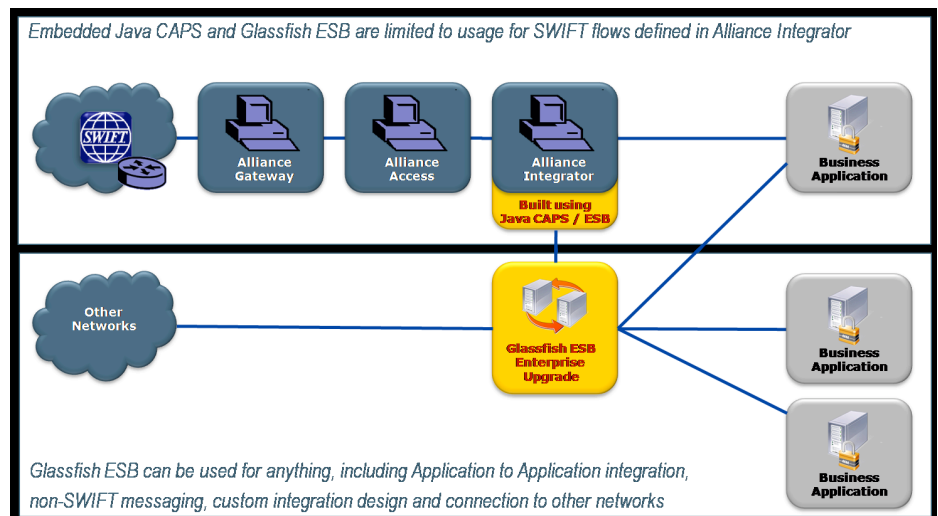
Alliance Integrator and Sun's Java CAPS, with GlassFish ESB, are complementary toolsets. Where Alliance Integrator is very focused on providing top-notch support for SWIFT flows, Java CAPS and GlassFish ESB provides an open integration platform for any system and any message flow.

A SWIFT integration hub, usually part of or linked to a payment processing hub, is best served with a combination of Integrator's SWIFT focus and GlassFish ESB's openness.

Furthermore, Alliance Integrator, GlassFish ESB, and Java CAPS work perfectly together.

SWIFT participants as well as SWIFT partners will benefit from an Alliance Integrator installation that has been extended to an enterprise no-limits GlassFish ESB installation. The combination ensures that the integration requirements of SWIFT projects will be covered with the technology infrastructure. SWIFT-specific integration logic such as connectivity to the SWIFT network through Alliance Access, configurable workflows for SWIFT Solutions, access to FIN, InterAct and FileAct, SWIFT messaging, mapping, validation, and reconciliation are covered by Alliance Integrator. Application-to-application integration, custom integration designs, non-SWIFT message flows and connections to other networks are covered by the GlassFish ESB extension. Since the technology used to build Alliance Integrator includes GlassFish ESB, there are no grey areas.

When installed simultaneously, the Alliance Integrator and GlassFish ESB environments can be separated, so that while testing the environments it's clear whether to call SWIFT support for Alliance Integrator or Sun support for the GlassFish ESB environment.



Sun's Alliance Integrator Extension Program

Sun's Alliance Integrator Extension Program offers a smooth, risk-free way for Integrator customers and partners to complement Integrator's rich functionalities with the openness of Sun's GlassFish ESB software.

For any firm that has acquired a standard Alliance Integrator license, Sun offers an extension program, extending Alliance Integrator to allow for enterprise-wide integration, for a 3-year or 5-year subscription license cost that is equal or lower than the one-time license and recurring maintenance cost of Alliance Integrator over the same period of time. The extension program offers an extension license as well as an extension training.

- The Alliance Integrator Extension Software Package is designed to fulfil the most common integration tasks required on top of Alliance Integrator, and is based on GlassFish ESB.

- The Alliance Integrator Extension License is designed to match the bands that are used for the Alliance Integrator pricing and makes it easy to predict TCO.
- As well as an extension license, Sun provides an extension training to GlassFish ESB that allows you to get started easily.

The Alliance Integrator Extension Software Package

To firms and partners who have licensed Alliance Integrator from SWIFT, Sun is offering an Extension Software Package that consists of the following components.

For a more detailed description of the below components, please refer to Appendix B.

- **GlassFish Enterprise Server**

The Sun GlassFish Enterprise Server (formerly Sun Java System Application Server) is a comprehensive support offering for GlassFish, the leading open-source and open community platform for building and deploying next-generation applications and services.

- **NetBeans™**

An Integrated Development Environment for software developers. You get all the tools you need to create professional desktop, enterprise, Web, and mobile applications.

- **Gold Support**

24x7 phone support with SunVIP Interoperability Support and Live call transfer.

- **BPEL Service Engine**

A WS-BPEL 2.0 compliant engine that provides business process orchestration capabilities.

- **Java EE Service Engine**

Makes the whole EE environment available as a JBI component so that EJBs can communicate with other JBI components through the NMR and vice versa.

- **XSLT Service Engine**

An engine for applying XSLT transformations with basic orchestration capabilities.

- **Data Mashup Service Engine**

Provides a single view of data from heterogeneous sources, including static Web pages and tabular data exposed as Web services. Provides options to join/aggregate/clean data, and generate a response in a WebRowSet schema.

- **HTTP Binding Component**

A JBI component for sending and receiving messages over HTTP.

- **File Binding Component**

Polls for files on the file system, and provides for writing files to the file system.

- **JMS Binding Component**

Receives messages from and sends messages to a remote service through JMS. Also supports WebSphere MQ/MQ Series.

- **JMS JCA Adapter**

A JCA Resource Adapter for accessing JMS Providers from EE or Java programs.

- **LDAP Binding Component**
Reads messages from and writes messages to an LDAP server.
- **FTP Binding Component**
Polls for files on an FTP server, and provides for uploading files to an FTP server.
- **Database Binding Component**
Reads messages from and writes messages to a database over JDBC.

The Alliance Integrator Extension License

The license structure for the above package, including Gold support depends on the band in which the customer operates, as defined by SWIFT.

- **Low Volume Customers**
Daily volume lower than 2,000 messages
- **Medium Volume Customers**
Daily volume between 2,000 and 50,000 messages
- **High Volume Customers**
Daily volume higher than 50,000 messages

As opposed to Alliance Integrator — which offers a one-time license with maintenance upgrades, the Sun Alliance Integrator extension program offers a subscription license.

The license allows for usage of the above GlassFish ESB package in development, test and production environments, as well as cold backup servers. There is a limitation on the number of servers in the production or hot backup environment as indicated in the table below.

The following price structure is valid as from April 2009, and can be changed without prior notice. Please get a confirmation through SWIFT@sun.com.

GlassFish ESB Upgrade Price			
	3 year cost (USD K)	5 year cost (USD K)	Server deployment limitation
Low volume	\$75	\$100	1 physical server or 4 virtual servers for production environment
Medium volume	\$75	\$100	1 physical server or 4 virtual servers for production environment
High volume	\$150	\$200	2 physical servers or 8 virtual servers for production or hot backup environments

The Alliance Integrator Extension Training

The Java CAPS and JBI Extension Training allows you to confidently complete any SWIFT or non-SWIFT integration project.

The Java CAPS and JBI Extension Training is designed and organised by Sun to complement the three-day Alliance Integrator training offered by SWIFT. In only two additional days, this training presents Sun's offering, discuss the fundamental concepts behind Java CAPS and Integrator co-existence and provides intensive hands-on experience with Java CAPS and JBI.

The trainers are experienced Java CAPS and JBI consultants. They will be able to provide best-practice advice and personal experience.

For the course agenda, please refer to Appendix A.

More Information

For more information about the program, the extension license model or the extension training, please contact Sun at SWIFT@sun.com.

Appendix A

The Alliance Integrator Extension Training

Target Audience

- IT people who will be actively involved in the implementation and integration project based on Alliance Integrator
- IT architect, developer, integrator, support engineer, and designer

Prerequisites

- Have attended the Alliance Integrator training from SWIFT
- Have experience handling NetBeans and Java programming knowledge

Course Schedule

- This course is given on special request only, please contact Sun through SWIFT@sun.com.

Course Content

- **Co-existence Strategies (one hour):** Lecture on how Integrator, Java CAPS, and GlassFish ESB can co-exist.
- **Modifying Environment Properties (one hour):** By the end of this exercise you will have used the command line interface to modify Environment Objects.
- **JMS and JCA (two hours):** By the end of this exercise you will have built, deployed, and unit-tested a simple message receive and send between JMS and File adapters.
- **Using Custom OTDs and XSDs (two hours):** By the end of this exercise you will have built and unit tested a User-Defined OTD and XSD that can parse a delimited .csv file that contains inventory records delimited by carriage-return, line-feed (`\r\n`).
- **BPEL 2.0 Persistence (two hours):** By the end of the exercise you will have enabled persistence for a BPEL 2.0 business process.
- **Monitoring: Repository, GlassFish, Queues (one hour):** By the end of this exercise you will have walked through monitoring and queue management.
- **BPEL 2.0 Projects (four hours):** This is an introduction to using BPEL 2.0 for projects and includes full development to deployment and testing.
- **Debugging in Repository and JBI (two hours)**
- **JDBC eWay for Persisting Data (one hour):** Lecture and demonstration.

Appendix B

Extension Software Package Detailed Description

GlassFish Enterprise Service Bus (ESB)

GlassFish ESB is a lightweight and agile ESB platform that packages the innovation happening with Project Open ESB, the GlassFish application server, and the NetBeans IDE into a commercially supported, enterprise-class platform.

GlassFish ESB runs on a wide variety of hardware and operating systems including Solaris, Linux, Windows, Mac OS.

Highlights:

- Provides a lightweight, modular architecture that enables agile SOA-based service development, deployment, and testing.
- Delivers a plug-and-play architecture where components and services can be integrated in a vendor-independent way.
- Promotes greater innovation and transparency through the open-source, ESB-community approach.
- Allows vertical and horizontal scalability via the asynchronous, decoupled design model of JBI-based component architecture.
- Provides your choice of development language, tooling, topology, and architectural style (MOM, SOA, EJB, BPM, EDA, and so on).
- Is part of the Sun GlassFish Portfolio, a cost-effective, open Web application platform that combines the best of open-source software and support in a single package.

NetBeans IDE

NetBeans IDE is an Integrated Development Environment for software developers. You get all the tools you need to create professional desktop, enterprise, Web, and mobile applications with the Java language, C/C++, and even dynamic languages such as PHP, JavaScript, Groovy, and Ruby. NetBeans IDE is easy to install and use straight out of the box and runs on many platforms including Windows, Linux, Mac OS X, and Solaris.

The NetBeans IDE 6.5 provides several new features and enhancements, such as rich PHP, JavaScript, and Ajax editing features, improved support for using the Hibernate Web framework and the Java Persistence API, and tighter GlassFish v3 and MySQL integration.

Highlights:

- **Java Desktop Applications**
Create professional desktop applications using the NetBeans Java GUI Builder with Swing Application Framework and Beans Binding support.
- **Java EE and Web Applications**
Build Web applications using Ajax, JavaScript, and CSS. Support for frameworks include JSF, Struts, Spring, and Hibernate. Full set of tools for EJB development.

- **Visual Mobile Development**

Create, test, and debug GUI applications that run on mobile phones, set-top boxes, and PDAs.

- **PHP Development**

A fast and light-weight PHP IDE with code completion and quick fixes, integrated FTP and Xdebug, and support for popular Web services.

- **Ruby and Ruby on Rails Development**

Powerful Ruby editor with code completion and debugger, and full support for Ruby on Rails. Includes the JRuby runtime.

- **C and C++ Development**

Full-featured C/C++ editor, debugger, project templates, support for multiple project configurations, remote development, and packaging of completed projects.

Gold Support

Gold Support for the GlassFish Portfolio products includes:

- Software Indemnification and Documentation Set
- Certified Extensions and Patches
- Unlimited number of incidents
- Unlimited number of Customer Support Contacts
- 24x7 Online and Telephone Technical Support
- Web-based Case Management
- Alert and Notifications
- VIP Interoperability Support
- Access to Knowledge Base
- Access to SunSpectrum™ eLearning Library

BPEL Service Engine

The BPEL Service Engine is a JSR 208-compliant JBI runtime component that provides services for executing WS-BPEL 2.0 (or simply BPEL) compliant business processes. The service engine supports one-way, request-response operations (as defined in WSDL 1.1), within stateful, long-running interactions that involve two or more parties. Asynchronous request-response is accomplished using two one-way operations, one implemented by a partner, the other implemented by the business process using correlation. Furthermore, an easy-to-use BPEL editor interface is provided to allow you to edit, build, test, and deploy your project to the BPEL Service Engine.

The BPEL Service Engine supports the following features:

- Standard JBI 1.0 engine component
- Supports BPEL 2.0, see BPEL 2.0 Language Constructs
- Provides and consumes Web services defined by using WSDL 1.1
- Exchanges messages in JBI-defined XML document format for wrapped WSDL 1.1 message parts

- Implements endpoint status monitoring
- Supports multiple-thread execution
- Supports debugging of business processes
- Supports database persistence of business process instances for reliable recovery from system failure
- Supports load balancing and failover when clustered

Java EE Service Engine

The Java EE Service Engine acts as a bridge between GlassFish and the JBI environment for Web service providers and Web service consumers deployed in GlassFish.

Features:

- EJBs/Servlets packaged as Web services and deployed on GlassFish are transparently exposed as service providers in JBI Environment.
- Java EE Components — EJBs/Servlets can consume services exposed in JBI environment using the Java EE service engine without being aware of the underlying binding/protocol such as SOAP, JMS etc. exposing the Web service.
- In-process communication between components of application server and JBI components to increase request processing speed.
- Any component that is plugged into ESB can directly communicate with Java EE applications. For example, clients of various bindings such as SOAP or JMS can communicate with Web services developed using Java EE via open-esb because of Java EE Service Engine.

XSLT Service Engine

The XSLT Service Engine is a JBI component capable of provisioning XSL stylesheets as Web services. Using a custom configuration file (described here), XSL stylesheets are mapped as service endpoints and deployed as JBI service units. An XSLT endpoint can even be chained, invoking other service endpoints as needed.

The XSLT Service Engine supports the following features:

- **XSL-based Transformation Processes:** a transformation process, described in detail on the TransformSL wiki page, is a list of activities executed sequentially. The supported activities are transform and invoke, though more may be added later.
- **XSL Parameters:** just as any XSL stylesheet can be supplied values for global parameters, so can any XSL stylesheet deployed as part of a transformation process. Parameter values can be taken from message exchanges involved in the process at runtime or specified as literal XML content in a file or in the transformation process itself. Please see the TransformSL Param wiki page for more information.
- **Multiple Invocations and Transformations:** as a by-product of incorporating the transformation process into XsltSE, there is no limit on the number of transformations applied or service endpoints invoked during a process' execution.

- **Limited Variable Support:** any message exchange as itself or by one its parts can be expressed using variable names. These variables are used as the source and result for transformations as well as indicating the input message for invoked service endpoints
- **All TransformSL variables follow the BPEL convention:** \$varName.partName. For more information, please see the variables wiki.
- **Calling Java extension functions that are packaged in the XSLT project as Jar files:** Calling Java extension functions.
- **Basic Fault Handling:** add Transform activities to apply to a fault message, which is then used as a reply. Please see the XsltSE Fault Handling wiki for more information.

Data Mashup Service Engine

The goal of this engine is to provide an implementation of JBI compliant data mashup service engine which will give a single view of data from heterogeneous sources within the enterprise with ability to source data from static Web pages and tabular data exposed as Web services, join/aggregate them, cleanse the data and generate a response in a WebRowSet schema.

Features:

- Integrating Information from heterogeneous sources viz. Relational databases, flat files, DCOM documents, spreadsheets, XML, HTML, RSS/Atom, Xquery RowSet to provide unified view.
- Creating a Data Mashup Services capability in SOA (JBI based) using OpenESB and NetBeans Enterprise pack infrastructure.
- Exposing the Aggregation of Enterprise Data Sources to Mashup Client frameworks, thus an enabling technology for Web 2.0 style applications in the enterprise.
- Time bound View Caching for improved response times, one can call this virtual materialized view.
- Composite weaving pattern can be applied to provide multiple views of the response through XSLT transformation.
- Extensibility through the ability to consume JBI Services.
- Transforming the response to various formats by weaving the output with an XSLT Service Engine; hence enabling multi-channel deployment.
- Reuse NetBeans Database Explorer to browse source tables; Drag-n-Drop these tables into the Mashup Editor to define the join conditions.
- Ability to view the ResultSet using the Mashup editor, Mashup View Cache Management.
- Creating a DataServices Layer in a true Service Oriented architectures.
- Enabling Integration on-demand.

HTTP Binding Component

The HTTP Binding Component provides external connectivity for HTTP and SOAP over HTTP in a JBI 1.0 compliant environment. The Component enables external components to invoke services, hosted by the JBI platform, using SOAP/XML messages over the HTTP/HTTPS protocol. It also allows JBI components to invoke external Web services using the same SOAP/XML over HTTP/HTTPS protocol.

Features of the the HTTP/SOAP Binding Component include the following:

- Supports the WSDL 1.1 and SOAP 1.1 specifications. Message exchanges to and from this binding component make use of the JBI WSDL 1.1 wrapper for the normalized message.
- Implements HTTP and SOAP binding as defined by the WSDL 1.1 specification.
- Follows WS-I 1.0 conventions and adds additional support for non-conforming components.
- Supports Document and Remote Procedure Call (RPC) style Web services.
- Supports literal and encoded use.
- Supports the common convention of WSDL retrieval via <service uri>?wsdl.
- Uses XML Catalogs following the OASIS Committee Specification, which allows the component to resolve schemas locally without resorting to network access.
- Supports JBI service unit deployments to define the Web services to provide or consume.
- Makes use of the WSDL extensibility (standard HTTP and SOAP extensions) to define external communication details for the Web services to provision or consume.
- Supports Binding Component to Binding Component Connection, which allows direct HTTP Binding Component to HTTP Binding Component connections within a composite application.
- Enhanced Environment Variable Support.

File Binding Component (BC)

File BC provides a comprehensive solution as a transport to interact with the file system within a JBI environment. Looking at what File BC does at a very high level: On the server side, File BC polls for inbound messages, stored in file(s), in a specified directory. On the client side, File BC puts messages into file(s) in a designated directory. The design time component of File BC is a NetBeans module that provides plug-in to NetBean's project system and thus defines how File binding can be used. The runtime component implements all required component interfaces in JBI specification and provides the functionality to act as a proxy to services enabled using the file protocol.

Features:

- Support in only, in & out, and out only message exchange mode and read on demand operation
- Support Application Variables
- Support Application Configuration
- Message exchange redelivery capability

- Message exchange graceful recovery
- Inbound throttling to control message handling concurrency
- Multiple Endpoints to poll the same directory
- Polling and writing multiple records per file
- Staging, achieving and error handling files
- Binary attachment support

JMS Binding Component

The JMS Binding Component provides a way to interact with message servers from within the JBI environment. JMS BC provides both inbound and outbound message processing. For inbound processing, JMS BC reads a message from the specified destination and forwards the message to a provider in JBI. For outbound processing, JMS BC sends a message to a destination. There is a NetBeans module to help work with JMS BC-specific binding configuration, and JMS BC runtime that implements all SPIs of JBI specification for a component.

Features:

- Handles text, XML, and binary data
- Inbound in-only message processing. Supports following concurrent mode: Sync and CC (Connection Consumer)
- Inbound in-out message processing
- Sending message to JMS server
- Support request-reply message exchange with JMS server
- Reading (on demand receive) message from JMS server
- Implements connection pool for resource sharing
- Support throttling to control flow of message
- Support for XATransaction (using appserver TransactionManager)
- Supports following message servers: Sun Java System Message Queue, JMS Grid, WebSphere MQ/MQ Series, WebLogic JMS, JBoss JMS, and Generic JNDI support.
- Support message recovery on failure
- Dynamic endpoint configuration. Provide support for overriding binding configuration dynamically.
- Message redelivery handling

JMS JCA Adapter

JMS JCA is a very feature-rich library that abstracts differences between JMS servers and provides a single interface to these JMS servers, and adds a number of additional features on top of this.

It consists of three parts:

- **JCA Container**

A stand alone JCA container consisting of a powerful connection manager that can be used in combination with a transaction manager. Although specifically developed for JMSJCA, it can also be used separately for other connectors.

- **JMS Adapter connecting to multiple JMS Servers**

A Resource Adapter that wraps JMS Client Runtimes of various JMS Providers. It can be used with the aforementioned JCA Container, but it can also be used with the JCA container in a J2EE application server.

- **Enterprise Integration Patterns Engine**

A number of Enterprise Integration Patterns, some of them found in the book “Enterprise Integration Patterns” book by Hoppe and Woolf.

LDAP Binding Component

LDAP (v3) is the Light weight directory Access protocol (RFC 3377). An Internet standard for accessing a directory (such as an address book) over TCP/IP. The Binding Component wraps the JBI interface. Users use this component to access operations that the LDAP provides to the open-esb world. LDAP Binding Component provides the services to add, search, update, and delete on LDAP directory. The Binding Component conforms to the JBI specification and enables LDAP server integration in a JBI environment. The LDAP component consists of two modules: run-time JBI module and design-time NetBeans plugin. LDAP Binding Component run-time module implements interfaces defined in the JBI framework and interacts with the external LDAP server for managing the directory services. The run-time uses LDAP v3 to interact with LDAP directory server. Design time NetBeans modules provides visual configuration of the binding component. It also defines how other components can integrate within the JBI framework. Users can update properties for the component and incorporate LDAP components with others in deployable service assembly.

Features:

- Provides the services to add, search, update, and delete on LDAP directory.
- Uses LDAP v3 to interact with LDAP directory server.
- Design time NetBeans modules provides visual configuration of the binding component.

FTP Binding Component

Binding components are used to send and receive messages via particular protocols and transports. They serve to isolate the JBI environment from the particular protocol by providing normalization and denormalization from and to the protocol-specific format, allowing the JBI environment to deal only with normalized messages – JBI Specification 1.0 (JSR 208).

Open ESB FTPBC is JSR 208 compliant, it provides a message transportation via FTP protocol so that services (which comprise operations) can be defined using WSDL and bound to FTP as its underlying message transportation protocol, other components in a JBI environment (for example, an SE can further orchestrate the services consumption and provision).

FTP BC implements all required BC interfaces in JBI specification so that it can be deployed and run in any JBI compliant target environment.

To facilitate the process of service definition and binding, the implementation also comes with a design-time component, a NetBeans module which makes WSDL authoring, and FTP binding a convenient process in NetBeans IDE.

Features:

- **Regular and Secured FTP**

FTP (RFC 959); Transport Security - FTP/TLS (RFC 2228 + RFC 4217) - Explicit SSL (RFC 2228 + RFC 4217) - FTP/TLS - Implicit SSL - FTP on SSL Socket

- **Connection Pooling**

- **Clustering Support**

- **Qualities of Services**

Component Logging Systemic Quality; Application Variable and Application Configuration; Runtime Monitoring (MBean); Re-Deliver; Throttling; Normalized Message Properties Normalize Message properties for OpenESB JBI components; FTPBC Defined Normalized Message Properties FTP BC Defined NM properties; Dynamic Addressing; Recovery (at least once delivery)

- **Message Type Enhancement**

Support Binary Message Content and Message As Attachment; Support Configurable Character Encoding for message content

- **Message Sequence Numbering**

Local Message Sequence Type 1; Transient Message Sequence

- **WSDL Extensibility Elements**

ftp:transfer; ftp:message; ftp:address; ftp:binding; ftp:operation

- **Synchronous Request-Response**

A file name tagging scheme that is implemented by the BC to help correlating the request with the response for a synchronous service invoke, user can enable by configuring an attribute value for one of the transfer extensibility elements (ftp:transfer, ftp:message)

- **Directory Listing Styles**

UNIX; NT 4.0; VMS; AS400; AS400-UNIX; UNIX (SJIS)

- **User Defined FTP Directory Listing Style**

Proxies (SOCKS4 and SOCKS5)

- **Pre and Post Operations**

DELETE - for post GET operation; COPY - for post GET operation; RENAME - for pre GET, post GET, and post PUT operation

Database Binding Component

The Database Binding Component (DB BC) provides a comprehensive solution for configuring and connecting to databases that support Java Database Connectivity (JDBC) from within the Java Business Integration (JBI) environment. Database BC is a JBI component that provides database operations as services. JBI components acting as consumers invoke these Web Services. The Database BC is an implementation in compliance with JBI Specification 1.0.

The Database BC supports the following database artifacts considered as Services:

- Table
- Prepared Statements
- Procedures
- SQL File

