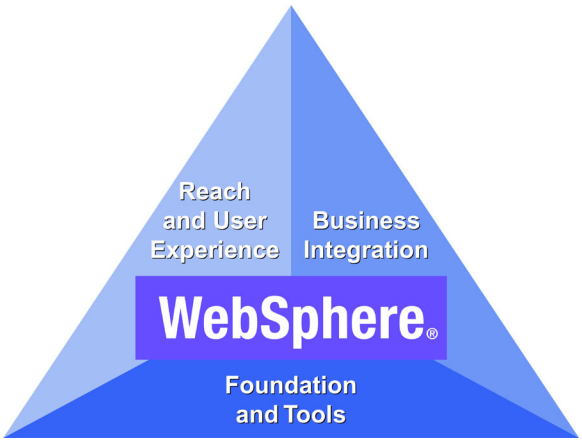
# Principles of dynamic e-business:

## Build- to-integrate

*WebSphere Application Server Version 5*

Jeff Reser
IBM WebSphere Application Server



# Comprehensive build-to-integrate platform

Being able to cleanly interoperate and integrate with new and existing applications and enterprise systems is a key success factor in building a dynamic and flexible e-business infrastructure. *IBM WebSphere® Application Server, Version 5* and IBM WebSphere Studio tools provide an approach to application integration based on open standards provided by J2EE and Web services. Using standards improves reuse of application components and simplifies integration. It also helps dissolve traditional barriers between application developers and integrators. WebSphere tools for developing business objects, business logic and integration logic are compatible, using one repository for metadata and a single approach for defining data handling and transformation.

Out of these premises for build-to-integrate comes a set of requirements that address the main business pain points toward an integrated and dynamic e-business environment.

*WebSphere Application Server, Version 5* fulfills these requirements through additional and enhanced support for services, open standards, connectivity, interoperability and messaging. Customers are looking for better ways to both smoothly integrate their businesses and manage sophisticated transactions. They want a way to update all critical resources inside of a transaction with high integrity. They want support for long-running transactions, with automated compensation in the event of a system failure. These are examples of deep transactional integration that provide both business and technical value.

## Enable dynamic application interactions

Enabling dynamic, flexible application interactions is a key attribute for those companies looking to build integration capabilities in their new application logic or to create new applications that are ready to be integrated with other systems, for example, companies looking to build-to-integrate.

A powerful part of integration is being able to build new applications that initiate and respond to asynchronous invocations, conversations, and broadcasts. *WebSphere Application Server Version 5* delivers extended services for mixed synchronous and asynchronous transactional environments as part of the native J2EE 1.3 and Web services environment. A comprehensive, self-contained Java™ Message Service (JMS) implementation is included that encompasses queue management and publish/subscribe components.

Java Message Service (JMS) relies on concepts established in the application messaging market for a number of years. JMS is based on the concept of a JMS consumer application (for example and Enterprise JavaBean or EJB components) listening for the arrival of a message on a queue, and then executing some business logic based on the message (for example updates to an online catalog), and possibly put a response (such as an acknowledgement) on a response queue. A new specification in J2EE 1.3, EJB 2.0 Message-Driven Beans includes the concept of a listener interface with the EJB container which monitors the appropriate queues.

*WebSphere Application Server, Version 5* adds value on top of the core EJB 2.0 specification by delivering container-managed messaging as a way of simplifying the development of these asynchronous applications. Container-managed messaging shifts responsibility for interacting with the messaging services to the EJB container, allowing EJB components to exploit messaging facilities without having to make explicit JMS calls. Container-managed messaging is

analogous to container-managed persistence, where data in entity EJB components can be persisted to the database tables without developers having to implement any database calls.

*WebSphere Application Server, Version 5* also implements a publish and subscribe message broker, where messages are published to a broker against an identifying topic. The messages are then distributed to subscribers whose registered subscriptions match the published messages. All of this is done within an environment that provides fault-tolerant clustering and load balancing, as well as full support for distributed transactions for a more robust implementation of JMS—offering high-performance publish and subscribe technology.

Another aspect of this paradigm is the utilization of a process manager that allows application interactions to be modeled from the perspective of the business process. The modeled business flow potentially invokes applications, and transactional application flows can be managed.

An example of dynamic application interactions is externally administering business policies. For industries such as insurance, there are many policies and many variable business practices that need to be maintained (such as defining the risk category for each policy holder). Many industries also need to respond quickly to regulatory and legislative changes, and doing this through programmer intervention can be prohibitively costly. What's desired is a framework for defining business policies that can be invoked from business logic. Imagine a world in which these same policies could easily be maintained and scheduled by business analysts. That level of flexibility would promote truly dynamic e-business. This flexibility is provided by *WebSphere Application Server, Version 5*.

Yet another aspect of flexibility is the capability to provide on-demand access to information that's needed by the business model. The ability to express queries for information in a way that efficiently maps to the business problem being solved is valuable. This represents another example for which *WebSphere Application Server, Version 5* delivers on the promise of dynamic e-business.

## Reuse and integrate disparate systems and applications

The ability to leverage existing systems and applications was a primary requirement since Web application servers were first created about five years ago. Integral to information and data connectivity is building new applications which integrate multiple back-end systems requiring data transformation and transactional integrity.

*WebSphere Application Server, Version 5* delivers a productive environment for visually creating dynamic application adapters that can be easily integrated with others within complex, multi-mode transactional schemes. The J2EE Connector Architecture (JCA) provides a consistent way of connecting to and communicating with a wide range of enterprise systems and applications, as well as advanced transaction coordination. This, coupled with XML based Web services, provide simple to sophisticated application connections and data aggregation.

The J2EE Connector Architecture defines function that *WebSphere Application Server* provides and which back-end system vendors (for example, IBM CrossWorlds®, SAP, PeopleSoft, Siebel, Oracle, and/or third-party connector developers) can use to plug into J2EE.

JCA has two basic components:

- **The Common Client Interface (CCI)** manages flow of data between the application and the back-end system and does not have any visibility into what the container and application server are doing.

- **Set of system-specific services.** *WebSphere Application Server* implements this as part of its base J2EE platform.

CCI is a programming interface that application developers and client programs can use to connect and access back-end systems. It is a low-level API and similar to Java Database Connectivity (JDBC). Unlike JDBC, however, CCI can work with non-relational systems. Although it 's possible for application developers to call the CCI directly, in most cases, an application developer will write to an abstraction layer, provided by the connector provider or enterprise application integration (EAI) framework vendor—simplifying the development process.

On the platform side, JCA defines a set of service contracts that a connector developer can expect will be available to the adapter at application runtime. The three services defined in 1.0 and implemented in *WebSphere Application Server, Version 5* include:

- **Connection management** enables *WebSphere Application Server* to create and manage connections to back-end systems. *WebSphere Application Server* also implements connection pooling, since connections to back-end systems are expensive. Connection pooling enables an EJB server to pool connections to back-end systems, so rather than opening connections on an as-needed basis, connections with data and services are established, configured, cached and reused automatically by the application server. This contract enables an application server to offer its own services for transaction and security management.

- **Transaction management** supports transactional access to underlying resource managers. This service enables the transaction manager provided within the EJB server to manage transactions across multiple back-end systems. Connector developers define what level of transaction support they need—for example, none, local (with a single back-end system and its resource manager) or XA, with either single- or two-phase commit—for working across multiple back-end systems and their associated resource managers.

- **Security management** enables the developer to define security between the EJB server and the back-end system. The specific security mechanism that is used is dependent on the security mechanism provided by the back-end system. For example, if a system requires Kerberos, then the connection developer will include it. Under the contract, the connector provider must also support user authentication, user authorization and any specific security contracts required by the back-end system.

## Unleash powerful Web services

A key set of market requirements for an integrating platform today involves merging Web services into the infrastructure. Web services provide another method of integrating applications and systems using de facto standards within self-contained software components. These software components can be described, published, located, and invoked over the Web. Web services perform encapsulated business functions from simple request-reply to full business process interactions.

Examples of these services include stock quotes and charting, credit card verification and payment processing, integrated travel planning and auctioning. Services can be mixed and matched to create complete end-to-end business processes, enabling dynamic integration with decreased human interactions.

New business opportunities can be created by exposing business and application services for integration by other businesses, organizations or platforms. Web services are pervasive and integral to an open cross-platform software strategy, key to dynamic business integration, and provide real business value within leading-edge integration solutions.

Many facets of application integration are evolving into Web services-based solutions, with *WebSphere Application Server, Version 5* leading the charge:

- Application integration provides friendly, intuitive interfaces using the latest Web services standards like Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL).
- Process integration is achieved using business rules engines and management systems powered by Web services.
- Information connectivity is solidified with prebuilt and tailored application adapters using the latest Web services standards for generation and service choreography.
- Access integration is expanded with flexible services in a network-aware business environment, with privacy and protection being provided for directories and registries such as UDDI (Universal Description, Discovery and Integration).
- The latest Web services standards for access and security are embedded in Version 5, along with native integration into the J2EE programming model.
- The Web services-based invocation framework provides protocol flexibility and easy-to-use tools that generate Web services applications that help you connect.

*WebSphere Application Server Version 5*, supports the latest Web services open standards. SOAP allows for easy, standardized access to public and private registries and other Web services applications. SOAP is the Web services version of Remote Procedure Call (RPC), an XML-based protocol and encoding format for inter-application communications. SOAP is widely viewed as the backbone to a new generation of cross-platform, cross-language distributed computing applications, termed Web services.

SOAP has evolved through several generations and the current spec, SOAP 1.1, is fast growing in popularity and usage. The third generation of Apache SOAP uses event-based parsing to achieve significantly greater speed than earlier versions of Apache SOAP. The Apache SOAP architecture, which is incorporated in *WebSphere Application Server, Version 5*, gives the developer complete freedom to insert extensions into the engine for custom header processing and system management. Apache SOAP defines a set of stable, published interfaces for component-oriented deployment. Apache SOAP also provides a clean and simple abstraction for designing transports (for example, senders and listeners for SOAP over various protocols—such as SMTP, FTP, message-oriented middleware), and the core of the engine is completely independent of the transport.

Finally, private UDDI registries offer protection and security for Web services applications and data access. There will be two basic models by which UDDI registries are used:

- Public registries enable anyone to publish their services and for other, effectively anonymous users to locate them and understand how to use them. This is not to imply that anyone can use the actual service. Normal access restrictions will still apply from a technical and business perspective. These are operated currently by IBM and Microsoft®, and will be shortly joined by HP and SAP.

- Private registries provided by *WebSphere Application Server, Version 5* can be used within the enterprise or between closed groups of trusted partners. In such an environment, you simply e-mail others the information required to use your Web services; However, UDDI provides a dynamic mechanism by which they can be published and consumed. *WebSphere Application Server, Version 5* provides support for both types of UDDI registries. The private UDDI implementation allows organizations to install an in-house private directory for the Web services applications.

*WebSphere Application Server, Version 5* also includes support for Web services gateways. These gateways offer more security and protection by filtering Web services access to registries and other applications.

## Choreograph application interactions

Being able to choreograph application interactions easily helps create real-time adaptive applications. With new *WebSphere Application Server, Version 5* features, new applications can be generated with adaptable intra-application flows and behaviors that can be changed dynamically via human interactions or rules engines.

As a result, businesses can maximize both programmer productivity and short-term return on investments by facilitating the reuse of development assets and automating the process of building, deploying, and managing applications. *WebSphere Application Server, Version 5* features an open approach to transforming any application asset into a modular component—accessible for reuse by other developers within an integrated development and deployment environment.

Choreographing and managing application interactions continue to be a strong market need in dynamic e-business infrastructures. Workflow can be we broken out into two types: macro and micro. While macro-flow management deals with applications, processes, servers, and systems—micro-flow management deals with controlling individual, synchronized, and asynchronous transactions.

*WebSphere Application Server, Version 5* addresses both flow models by controlling the distributed service flows from and to the Web application server environment. Flow management and state management are done within *WebSphere Application Server, Version 5*. In addition, WebSphere Studio offers visual tools that help create and assemble Web services applications through application usage profiling and a business rules engine that can be populated dynamically for controlling a Web services-oriented architecture—as well as EJB transactions. This all enables you to build new applications requiring flexible and real-time adaptable intra-application flows and behaviors.

Another key requirement that companies place on their Web Application Server is support for process automation facilities. The ability to host and schedule a living business model in the application server environment brings new opportunities for seamless process integration. This

is accomplished through service choreography and it includes scripted interactions with Enterprise Information Systems and other services as well as longer running workflows that tie together activities into more course-grained business process steps.

Related to this is support for business entities that adapt themselves to diverse business processes. Companies are looking for a cost-effective application-serving environment in which these process integration capabilities can be easily leveraged, with a single administrative and operational view of the runtime environment. This includes the ability to not only run business processes, but to also monitor them. *WebSphere Application Server, Version 5* introduces all the capabilities to provide effective service choreography for application interactions.



IBM Corporation
Software Group
Route 100
Somers, NY 10889
U.S.A.

Produced in the United States of America
06-02


This document contains information relating to future or anticipated releases of products and represents IBM's current intentions, goals and objectives. The information in this document is subject to change or withdrawal without additional or prior notice. This Product will be available in multiple configurations and for that reason not all functions discussed in this document are included in all configurations of this Product or will be available upon the initial release of a configuration of the Product.