WebSphere® software

Redbooks Paper

Chris Nott

Patterns: Using Business Service Choreography In Conjunction With An Enterprise Service Bus

Overview

This IBM® Redpaper briefly describes the characteristics of business process management (BPM) and then introduces some design concepts for using a business service choreography (BSC) engine in conjunction with an Enterprise Service Bus to implement BPM. It is assumed that the reader is familiar with the following topics:

- Service-oriented architecture and the Enterprise Service Bus
- Web Services
- Process Integration patterns from the Patterns for e-business
- Service-oriented architecture patterns

Relevant information about these topics can be found in Chapters 3 and 4 of *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346.

In this Redpaper, we extend the SOA patterns. We define the way in which Process Integration patterns in business service choreography design can be used in conjunction with an Enterprise Service Bus to form composite SOA patterns.

Business process management and business service choreography

In this first section, we will explore the characteristics of business process management. Then we will look at the characteristics of business service choreography and explore how it can provide capability as a component in a service-oriented architecture (SOA) to implement business process management. We will also assess the impact of introducing an Enterprise Service Bus (ESB) into the architecture.

We will standardize the term Business Service Choreography (BSC) because it is aligned with the Business Process Choreography Services component in the Application Layer of the on demand Operating Environment.

Characteristics of business process management

The readers of this Redpaper are likely to be IT skilled. Consequently, it is worth emphasizing that the goal is to capture design patterns to enable the management of business processes.

- As part of the evolution to an on demand business, organizations need to move beyond the sharing of data between employees, customers and business partners to the sharing of process definitions. Processes can reflect events which occur across the entire business value chain.
- Flexibility is achieved through the ability to change business processes rather than simply the ability to create them in a business process management solution.
- It is the business which owns business processes and defines their requirements. Indeed, it should be recognized that business processes *are* the business.

  Successful business process management solutions require business analysts who model and simulate business processes and events: they cannot be defined by IT architects alone.
- Important business drivers for business process management are often related to tracking performance in terms of increasing profit, growing revenues and the ability to respond to changing market conditions.

  Regulatory bodies are imposing increasing demands for corporate governance. These can be facilitators of business process management because of the associated time scales for showing compliance.

  Businesses define their performance indicators required in solutions. Coupled with the ability to change processes, new business opportunities and efficiencies can be exploited.

- Business processes are about business functionality. Applications can be used to automate activities within these processes. In this context, applications are decoupled from business processes. This enables organizations to concentrate on business functionality rather than applications.
- The scope of business processes can extend beyond systems integration. It can include activities which are not related to systems at all.

Business service choreography

In general terms, business service choreography is about the development and execution of business process flow logic, which is abstracted from applications. Inherent in this are rules which govern the sequencing and control of service invocations, which in turn support these business processes and workflows.

Business Process Execution Language for Web Services (BPEL4WS) is an emerging standard for describing the process flow. It is described later in this Redpaper. By using it with Web services standards and technologies:

- Processes can be composed from the operations supplied by Web services. Access to existing applications and new components can be gained using an open standards based communications infrastructure.
- Interfaces to the processes themselves can be provided as Web services.

The separation of the business process flow from the implementation of the underlying Web services facilitates flexibility to adapt to changing business conditions. This is because:

- Service-oriented architecture encourages the encapsulation of service functions and the loose coupling of service interactions.
- WSDL enables the creation of explicit, implementation-independent descriptions of service interfaces.

Whereas the implementations of Web services themselves are process flows, reuse is promoted and increasing levels of abstraction are provided.

Spectrum of service choreography

In modeling process flows, there is a spectrum of service choreography which should be considered. This is shown in Figure 1 on page 4. At one end are the fine-grained activities required to perform a sequence of technical interactions, for example, integrating with legacy applications through green-screen interfaces. At the other end are end-to-end business processes which will often reach beyond the enterprise.

An example of such a business process is the placing of an order by a client and the control of its fulfillment, shipping and billing in conjunction with business partners. In the middle of this process, there might an activity which requires business service choreography to create the order record correctly in an ERP application through a sequence of interactions with it.

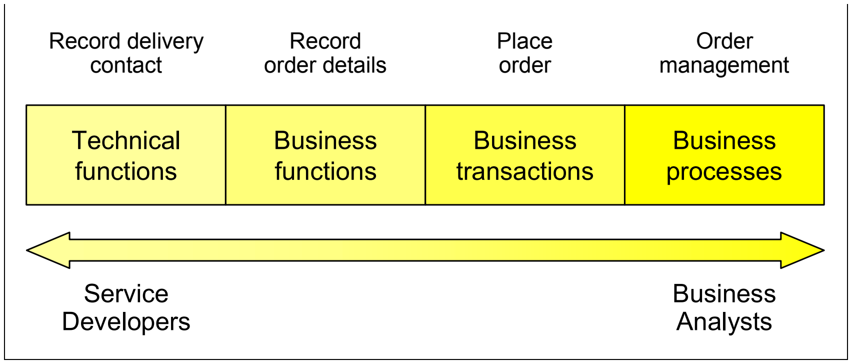


Figure 1 Spectrum of service choreography

Along this spectrum, there are different types of flows and each requires different types of skills to design and implement. Designing the interactions to create an order in an ERP application requires different skills, from modeling a business process to placing and fulfilling a customer order. This implies that the implementation of business service choreography may require multiple tooling capabilities.[1]

It is also possible that different execution mechanisms for service choreography could be used. In particular, technical functions which choreograph access to service providers can be implemented within the ESB component rather than in a separate service choreography component.

Fortunately, Web services are able to support these multi-level requirements because they can support a wide range of service granularity independent of the choreography.

Capabilities required for business service choreography

Definitions of the capabilities required for business process management and process integration are documented in many places, and some are included in the references listed. In relation to service choreography, the capabilities required for both development and execution of process flows include the following:

[1] Some concepts described here are attributed to reference [5].

- Support of process definition standards.
  For example, when importing a process definition which has been exported from a process modeling tool, conversion of the process definition may be involved. Native execution of process definitions can also reduce process failures.
- Use of standard transports and messaging to provide connectivity to services.
- Interoperability of processes through service interface definitions.
- Support for multiple levels of process abstraction.
- Monitoring and analysis of processes. This includes capturing information about process execution for historical analysis, integration with systems management and administration tools, logging and auditing.
- Scalability and performance of the infrastructure are necessary to meet required throughput and cycle times of non-interruptible processes (and sub-processes).
- High availability and reliability of the process engine and its underlying infrastructure is a key factor in making overall process flow execution reliable.
- Quality of service to provide transactional support, such as compensation. Restoration of failed processes to their most recent consistent states must be possible through recovery; many process flows can run over a long period of time.
- Security including authentication, authorization, non-repudiation and role-based access control.
- Flow control, including rules to control sequencing of activities, branching, parallel branch execution and recomposition.
- State management.
- Correlation of events or incoming messages with existing process instances.
- Public and private models for B2B.

Applying business service choreography in an SOA

The following benefits apply when using business service choreography in conjunction with an SOA:

- Coarse-grained services will typically promote the following benefits:
  - Tolerance of changes to business processes
  - Facilitation of reuse
  - Simplification of the coupling between services through choreography

  See Section 3.2.3 in *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346 for further discussion.

- There is loose coupling between different services. Again, this allows processes to change quickly.
- Separation of business flow logic from application and functional logic can be achieved, promoting flexibility.
- Implementation of functional services can change without affecting the sequence of activities in the process logic. Functional services expose business logic from packaged applications and legacy systems.
- Changes in business processes over time are facilitated. For example, the smooth integration of automated functions in place of manual activities.

Using business service choreography with an ESB in an SOA

An Enterprise Service Bus (ESB) provides a set of infrastructure capabilities, implemented by middleware technology, that enables the integration of services in a service-oriented architecture. The ESB has a single point of management over distributed mediation capabilities between service requestors and service providers. These providers may include functional services provided by existing applications.

The ESB is an infrastructure component and as such does not process business logic. As discussed in Section 4.3.2 of *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346, there are areas of lower-level logic which are difficult to categorize and can reasonably be included in the implementation of the ESB.

When considering business service choreography in conjunction with an ESB, the following distinctions can be made.

- Business service choreography capabilities supporting flow development and execution are separated from the ESB's mediation capabilities.
- From a business point of view, the ESB is stateless. State can be managed as part of service choreography.

  It is valid for an ESB to hold state when it invokes external services in certain cases. It may need to hold contextual information relating to timestamps, security and transactions, for example, and then reassociate the context with the inbound response from an external service provider.
- The separation of business flow logic from application and functional logic is extended with an ESB to location independence because there is greater decoupling between a service requester and service provider.
- Process logic can both invoke ESB services and be the implementation of services provided by the ESB. Thus, process nesting can be provided using

multiple service interfaces (recursively). Performance, however, should be a consideration in such a design.

By considering business service choreography services as a component, we can position them as relative to the ESB component in a service-oriented architecture. There are other commonly occurring components whose roles can be positioned relative to these:

- The ESB Namespace Directory, providing information to allow the routing of service interactions.
- The Business Service Directory, providing a taxonomy and details of available services to systems participating in a service-oriented architecture.
- The ESB Service Gateway, used to provide a controlled point of external access to services.

Figure 2 illustrates these components interacting with the ESB in an SOA-enabled infrastructure.

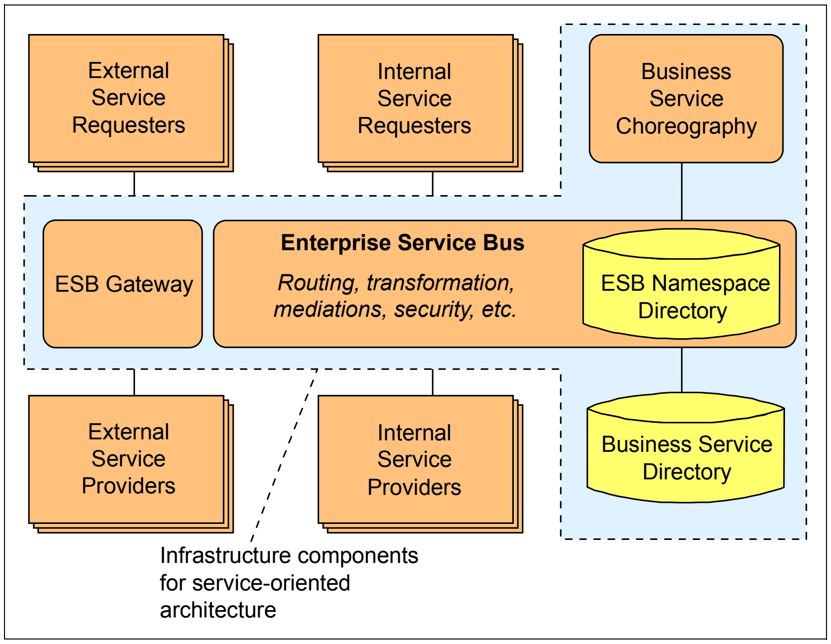

External Service Requesters
Internal Service Requesters
Business Service Choreography
ESB Gateway
Enterprise Service Bus
Routing, transformation, mediations, security, etc.
ESB Namespace Directory
External Service Providers
Internal Service Providers
Business Service Directory
Infrastructure components for service-oriented architecture

Figure 2 SOA enabling infrastructure components

Business service choreography and the Process Integration patterns

The business and IT drivers for the Patterns for e-business Process Integration patterns are as follows:

- The business process needs to be integrated with existing business systems and information.
- The business processes need to be integrated with processes and information that exist at partner organizations.
- The business activity has a need to aggregate, organize, and present information from various sources within and outside of the organization.

The early SOA patterns have been built from composite Process Integration patterns and are defined in *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346.

Process Integration patterns for business service choreography

It is clear from the business and IT drivers that the Process Integration patterns will apply to BSC. For business service choreography, the requirements are likely to demand the use of the more advanced Process Integration patterns. These are:

- Serial Process pattern (with Serial Workflow variation)

  The Serial Process pattern facilitates the sequential execution of business services. It enables the choreography of a serial business process in response to an interaction initiated by the source application. It improves the flexibility and responsiveness of an organization by externalizing process logic from individual applications.

  The Workflow variation extends the pattern to support human interaction for completing certain steps reflecting roles within the organization structure. This makes the process interruptible and provides support for long-running transactions.

- Parallel Process pattern (with Parallel Workflow variation)

  The Parallel Process pattern extends the serial service choreography capability provided by the Serial Process pattern by supporting parallel, concurrent execution of the sub-processes. By executing portions of the process flow in parallel, overall cycle times can be reduced.

Additional complexity is introduced with this pattern which spans process design, testing and operation:

- Start and join conditions must be defined for the sub-processes executing in parallel.
- Sophisticated runtime engines are required to manage the initiation and joining of parallel threads of control as an overall unit.
- Error scenarios must be carefully analyzed and managed.

As for the Serial Process pattern, the Workflow variation extends the pattern with human interaction and support for long-running transactions.

These patterns are shown in Figure 3.

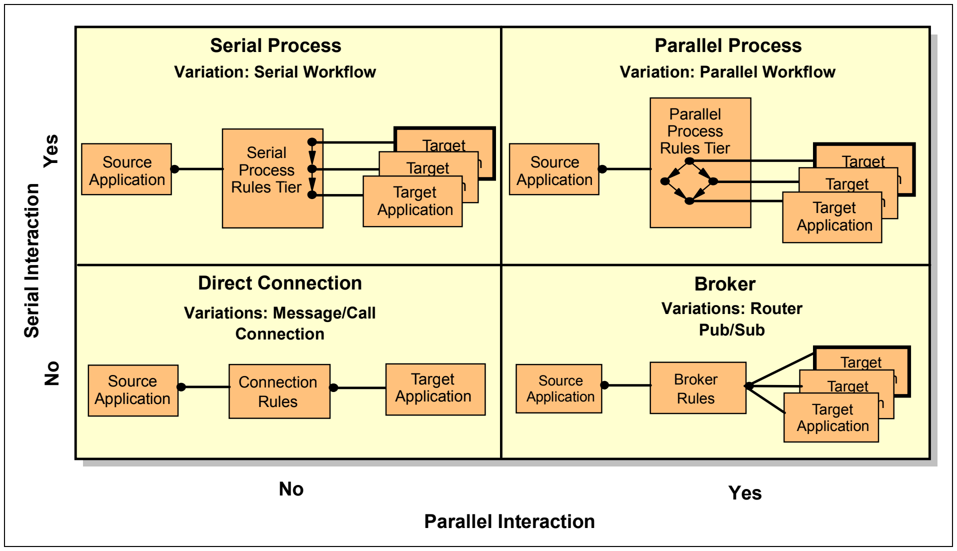


Figure 3 Process Integration patterns

Note that these Process Integration patterns can also be used to define functional services which can be invoked over the ESB; they can also provide an implementation of a service delivered by the ESB.

Further information about these patterns can be found in *Patterns: Serial and Parallel Processes for Process Choreography and Workflow*, SG24-6306.

The Process Integration::Zone pattern

The Patterns for e-business Process Integration patterns describe a Zone as an area in which a specific set of services is available. A Zone is accessed through one or more Gateways, which are also referred to as Ports in the SOA profile.

Examples of a Zone and Gateway might be:

- An intranet, accessed via a firewall and proxy server
- A J2EE Web container accessed through an HTTP listener
- A message broker accessed through a queue
- An Enterprise Service Bus accessed using a suitable protocol

In Figure 4, the Process Integration::Zone pattern is specialized using a Port as a Gateway to connect to the Business Service Choreography Zone.

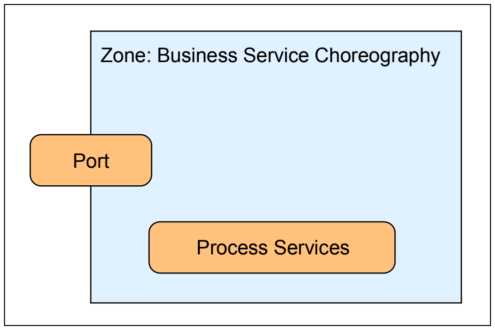


Figure 4 A specialization of the Process Integration::Zone pattern

We will now formally define the Business Service Choreography pattern.

Business Service Choreography pattern

The Business Service Choreography pattern occurs in designs where it is necessary to choreograph intra-enterprise and inter-enterprise services into business processes where each activity in the business process is defined as a service. The choreographed business process itself is also exposed as a Web service.

The business processes that are implemented in an enterprise typically require a mixture of human and IT resources. A business process can also be event-driven. For example, it can be paused to wait for an event and then resumed when a message arrives.

This pattern describes business processes that can be:

- Long-running (macro-flow) and interruptible (requiring human intervention)
- Short-running (micro-flow) and part of a one-business transaction

Next, we consider the drivers.

Business and IT drivers

The primary business driver is to support the composition of end-to-end business process flows by choreographing business services implemented by a number of target applications. This might be in pursuit of the following objectives:

- Improve organizational efficiency
- Reduce the latency of business events
- Support automated coordination of the business process flow
- Reduce cycle time through parallel execution of portions of a process flow
- Support human interaction and intervention within the process flow

As such, the pattern is required to allow multiple automated business processes to be combined to yield a new business offering or to provide a consolidated view of some business entity by integrating multiple corporate business systems.

From an IT perspective, the key driver for this pattern is to improve the flexibility and responsiveness of IT by externalizing the process flow logic from individual applications.

A robust and manageable service choreography infrastructure consistent with the principles of service-oriented architecture must also be provided.

Solution

We can define the Business Service Choreography pattern as follows:

- The Business Service Choreography pattern allows business service choreography capabilities to be made available to service interactions.
- The Business Service Choreography pattern exposes a set of Ports to service requesters to provide access to flows as services, both to initiate them and to resume them. Each Port is identified with a specific protocol and set of addresses through which it provides access to the Business Service Choreography Zone.
- The Business Service Choreography pattern uses a set of Ports to integrate with service providers. Activities in a process flow may consume a service through a Port which supports a specific protocol and a set of addresses specific to the service being invoked.
- The Business Service Choreography pattern contains a Process Manager that applies the capabilities for business service choreography to service interactions between service requester Ports and service provider Ports. These processing units will be instances of the appropriate Process Integration patterns to manage required business process logic across the spectrum shown in Figure 1 on page 4.

Note that the Workflow variations of the patterns described above allow for human interaction where tasks can be routed to human actors for completion.

This definition of the Business Service Choreography pattern expresses the access by such human actors as services.

The Business Service Choreography pattern is illustrated in Figure 5. It shows the Process Manager, which might be an implementation of the Serial or Parallel Process patterns (with or without the Workflow variation). Access to the Process Manager is through Ports. Three types are shown and are defined as follows:

- Receive

  The Receive Port waits for a message to arrive over a specific protocol to invoke a service which either initiates a process instance or allows an existing process instance to resume. In the latter case, the process instance is identified using correlation information provided by the requester. If a response is not required then the service operation is defined as one-way.

- Pick

  The Pick Port waits for one of multiple messages to arrive, allowing different courses of action to be taken in the process instance based upon the incoming message. The message either initiates a process instance or allows an existing process instance to resume based on correlation information. The service operation for a Pick Port is defined as one-way because a response is not required.

- Invoke

  The Invoke Port enables a request to be made using a specific protocol and a set of addresses specific to the service being invoked. The request may be a one-way or request-response operation.

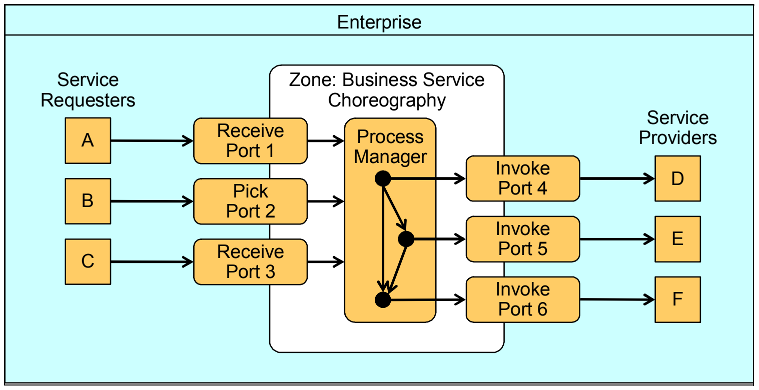


Figure 5 Level 1 decomposition of the Business Service Choreography pattern

Note: The illustration of a parallel process flow in Figure 5 is only used to show a flow. This could just as well be a serial process flow or an application, or indeed another Process Integration pattern.

In discussing integration problems and solutions, we often need to show components of the solution at different levels of detail. In the Process Integration patterns, we call these *levels of decomposition*. When a component is shown without any internal detail, we call this a level 0 representation. Subsequent decompositions may be labeled level 1, level 2 decompositions and so forth. This is intended strictly as a suggestive notational convenience; the different level numbers are helpful in sequencing diagrams but they have no absolute meaning.

Figure 6 shows the level 0 decomposition of the Business Service Choreography Zone, equivalent to the level 1 decomposition shown in Figure 5 on page 12.

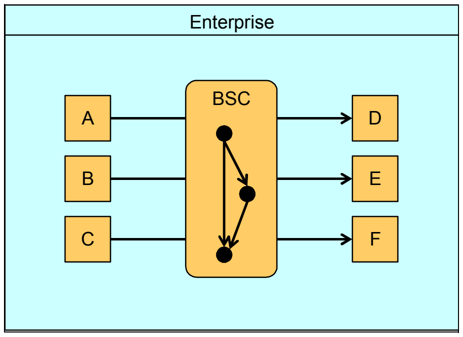


Figure 6 Level 0 decomposition of the Business Service Choreography pattern

Guidelines for use

The Business Service Choreography pattern can be employed when access to and access by choreography services is required. In the definition of the Zone, this access is through Ports. Each Port represents access to one or more addresses over a single protocol. This supports the following requirements for use in a service-oriented architecture. Using the Ports:

- A service requester accesses a flow using an interface which is independent of the flow implementation. The flow and content are encapsulated logic.
- A service provider is accessed by a flow through a Port using an interface which is independent of the provider's implementation. The provider encapsulates its logic.

Within the Zone itself, the Process Integration patterns can be used. It is likely that the Serial Process and Parallel Process patterns will apply to most scenarios.

Business service choreography may involve state management. Careful consideration needs to be given to how state is managed in a service-oriented architecture to guard against failures in a distributed environment. Connectionless service interactions where all the business data is defined in the interface should be used. This means that in a shared sequence of service interactions, each interaction should be uniquely identified, using a customer ID, for example. Therefore, the identify forms part of the service call. Explicitly sharing a process definition can further simplify the process of achieving this goal.

The Business Service Choreography pattern is able to support a range of service granularities. The pattern can be used in a service-oriented architecture to provide support across the entire spectrum of service choreography shown in Figure 1 on page 4. It is worth recalling two widely held views:

- Coarse-grained service definitions lead to greater flexibility.
- The use of simpler interfaces leads to more interface stability.

Therefore, well-defined, coarse-grained service providers are typically best suited for use with the Business Service Choreography pattern.

By taking appropriate design decisions in defining business services, the Business Service Choreography pattern can facilitate the automation of business processes. This is a significant step towards achieving the eventual goal where business users are able to create and change flows with minimal involvement from IT. In pursuit of this goal, it is necessary to monitor and measure the business effectiveness of the flows running in the zone.

Benefits

The Business Service Choreography pattern improves the flexibility and responsiveness of an organization by implementing end-to-end process flows and by externalizing process logic from individual applications.

These flows can provide automated straight-through processing or involve human workflow. The pattern is able to pass out exceptions from automated processing for manual intervention. By employing the Process Integration patterns, further flexibility is introduced by externalizing task-resource resolution rules.

Parallel execution of multiple tasks in a process can also reduce overall cycle times.

The pattern provides a foundation for automated support of business process management. This enables monitoring and measurement of the effectiveness of business processes.

Limitations

The primary purpose of the Business Service Choreography pattern is to coordinate processing. It is less concerned with providing the capability to allow the integration of information used by applications in terms of data propagation and replication. It is also not concerned with the aggregation of data from multiple data sources. Use of this pattern is not appropriate when integration of applications and data repositories is handled outside of any specific application or service request.

The Business Service Choreography pattern is not intended to be used to deliver IT (rather than business) functions and data to end users. Moreover, it does not provide the external interaction services for user and device integration.

Composite patterns for choreography with an ESB

This section illustrates how service requesters and service providers for the Business Service Choreography pattern can be accessed using an ESB. This will be done by describing composite patterns for using the Business Service Choreography pattern together with an ESB pattern.

In designing solutions, we will first consider the business process end of the spectrum of business service choreography.

A business-level solution will be focused on the modeling of full business processes, moving the required information between systems by making service calls. The model will typically choreograph coarse-grained services.

However, it is still valid for business transactions and business functions to be implemented using these composite patterns, where the services might be more fine-grained.

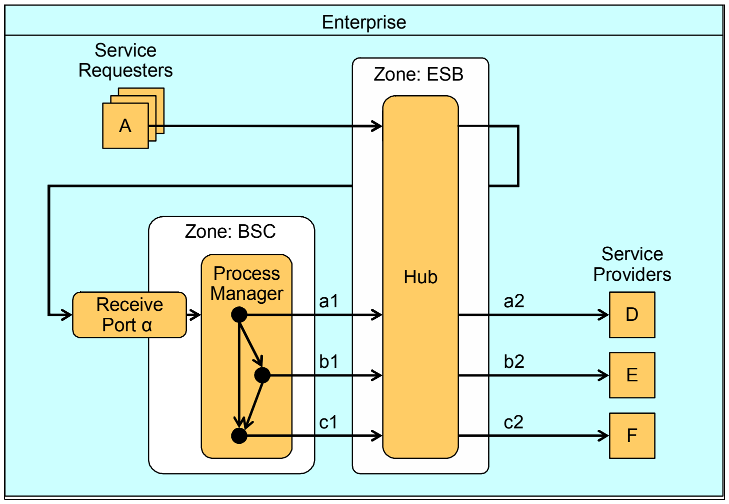


Figure 7 The business process exposed as a service by an ESB

Figure 7 shows a business process exposed as a service by an ESB as a composite pattern. It allows the process to be initiated by a service call made by a service requester. The Business Service Choreography Zone's Receive Port is listening to such service calls. Once instantiated, the activities of the business process can also be defined as calls to services provided by the ESB. The implementations of these services provided by the ESB and service providers are decoupled from and independent of the business process running in the Business Service Choreography Zone.

Figure 8 and Figure 9 on page 17 are alternative and logically equivalent representations of Figure 7. Notice that the two ESB zones shown in Figure 8 are one and the same; the names of the zones are also the same.

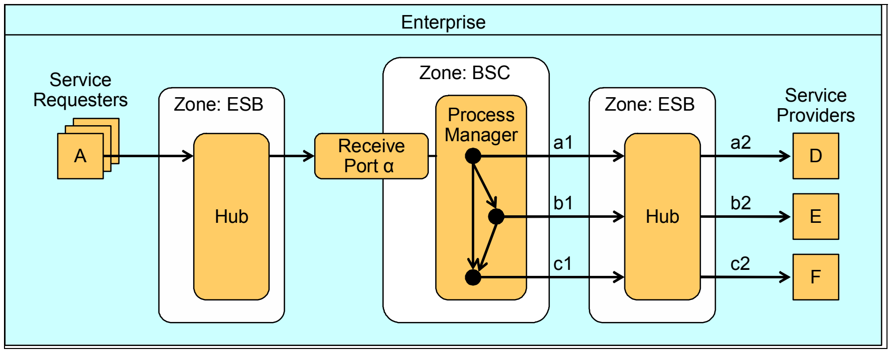


Figure 8 Business process exposed as a service by an ESB (alternative representation)

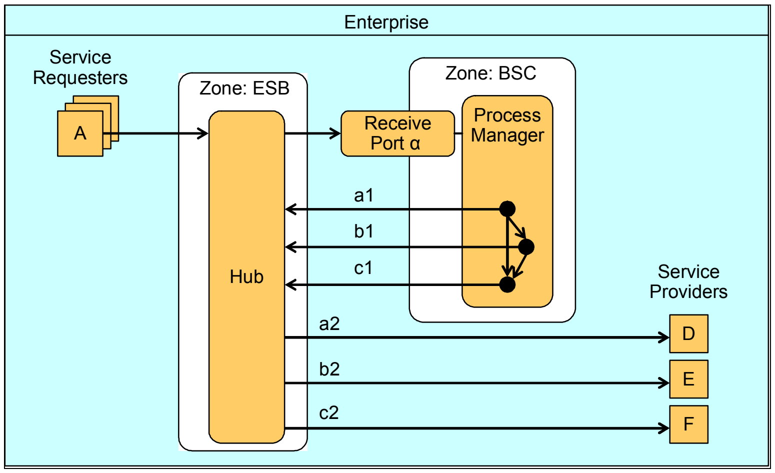


Figure 9 Business process exposed as a service by an ESB (alternative representation)

Figure 7 on page 16, Figure 8 on page 16 and Figure 9 show a partial view of the design. Figure 10 shows further detail in a full decomposition level 1 design.

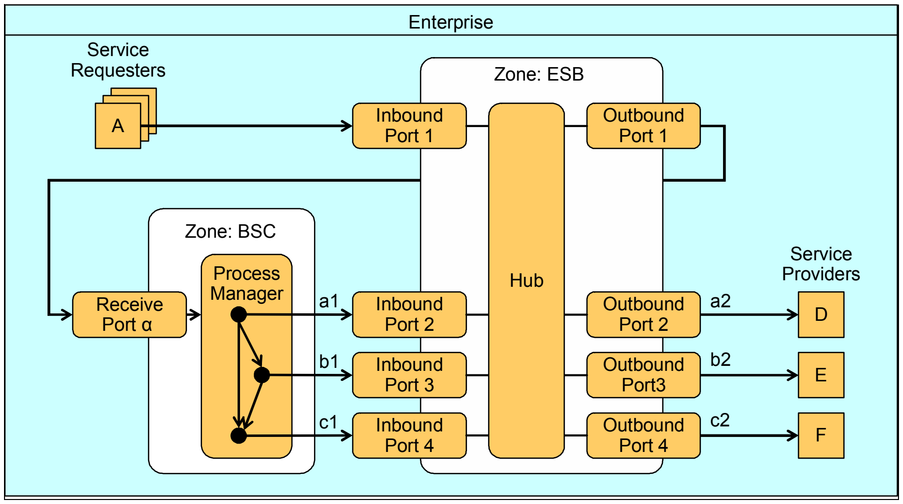


Figure 10 Level 1 decomposition of Figure 7 on page 16

Figure 11 on page 18 shows a more complex representation of the composite pattern which includes ports to allow the resumption of a suspended process. This allows a process to be interruptible. When interrupted, the process is waiting for a condition to occur which will allow it to be resumed. While this might be internal to the process (for example, a time-out expires), a message may be expected on a Pick Port. In this latter case, the message must be correlated with the process instance.

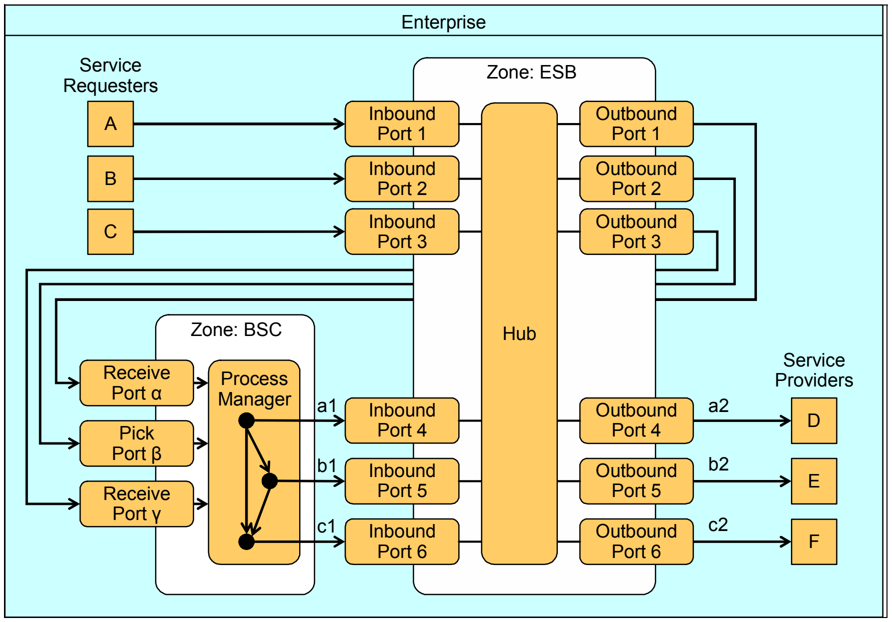


Figure 11 Using the ESB to initiate and resume business processes

While the discussion, so far, has largely focused on modeling business processes which can invoke services over the ESB, it is valid to use a process manager to define business or technical functions as fine-grained services and provide them to the ESB.

The composite pattern in which the Business Service Choreography Zone provides an implementation of a service delivered to the ESB is shown in Figure 12 on page 19. The service implementation is encapsulated in the white shape called ESB Service.

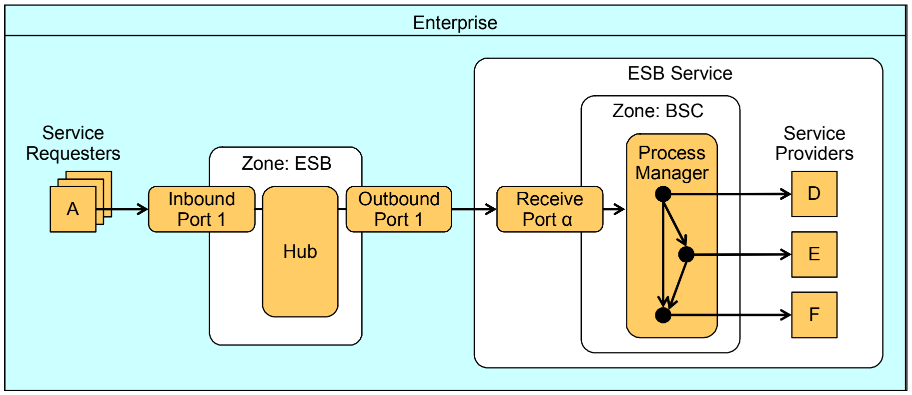


Figure 12 Using choreography to provide an ESB service implementation

An important design point to note here is that this composite pattern is topographically equivalent to those shown previously. In particular, notice that the instantiation of the business process shown in Figure 7 on page 16 is by service invocation from the ESB.

As shown in Figure 12, those flows which implement business or technical functions may invoke service providers directly by making direct connections. While the drivers for these finer-grained flows are typically for non-interruptible flows, this is not a requirement.

The invocation of service providers directly by a business process (rather than through the ESB) has the following implications:

- The ESB is not managing the service addresses of the providers and so they are not part of its single point of configuration management.
- There may be increased management and maintenance if these provided services are reused.
- The decoupling between the process and the service providers is reduced, which constrains flexibility.
- Performance may be improved.

By recognizing that a business process can be provided as a service on the ESB, we have a design pattern which allows fractal decomposition[2] to ever more fine-grained process services. This allows the separation of differently skilled process modelers across the spectrum of service choreography so that processes can be provided as differently grained services; the obvious limitation in applying this concept is performance degradation.

[2] Here, a process is a fractal: both compositions and decompositions of processes are processes.

In practice, organizations will have many existing applications which do not currently provide a standard service interface like Web services. As is the case with the ESB, direct calls using other technology options may be made from process flows in the Business Service Choreography Zone. These might include:

- WebSphere® Business Integration Adapters
- JCA
- RMI/IIOP
- JDBC
- Native WebSphere MQ (with no runtime JMS folder)

Figure 13 shows how business processes can be extended beyond the enterprise to interact with partner organizations. By applying the SOA composite patterns, we see that the introduction of an ESB Gateway allows the ESB to interact with other organizations over the Internet.

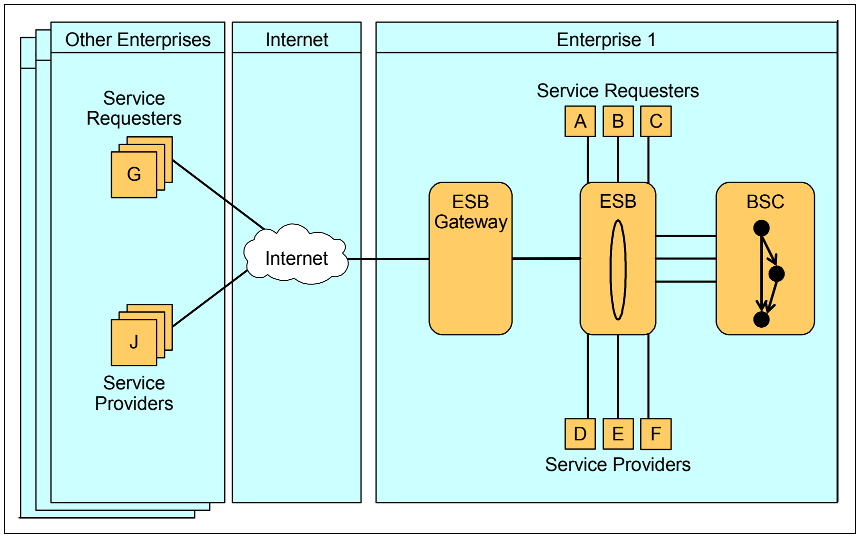


Figure 13 Using business service choreography in an extended enterprise environment

The introduction of this component allows specific additional quality of service requirements to be considered in the design process. These will be used to support the service-level attributes agreed upon among organizations as part of higher-level business processes.

Refinements using further levels of decomposition

The following figures show further levels of decomposition of the composite patterns we have discussed. The first of these, shown in Figure 14 on page 21, might be a direct refinement of the composite pattern shown in Figure 7 on page 16. In this level of decomposition, a router is assigned to each incoming and outgoing port. This signifies that each port has its own routing rules within a hub.

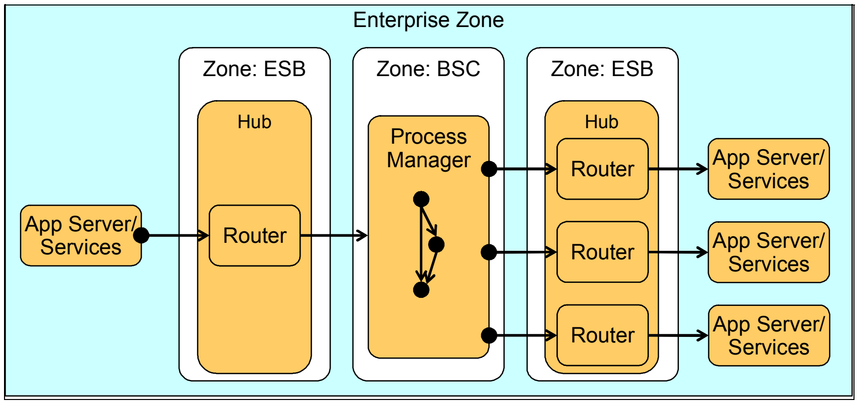


Figure 14   A refinement of Figure 7 on page 16

Figure 15 shows a refinement where choreography provides the full implementation of a service provided by the ESB. In this case, the activities within the flow make calls to application services using direct connections as opposed to invoking services provided on the ESB.

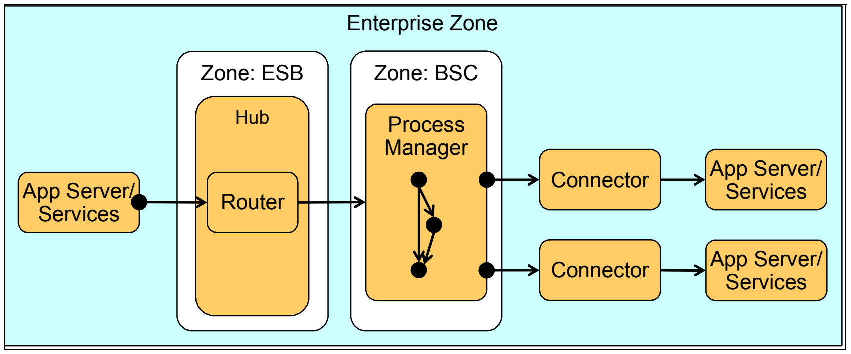


Figure 15   A refinement showing application access using connectors

Figure 16 on page 22 shows a combination of the concepts used in the previous two figures and illustrates a refinement with fractal decomposition.

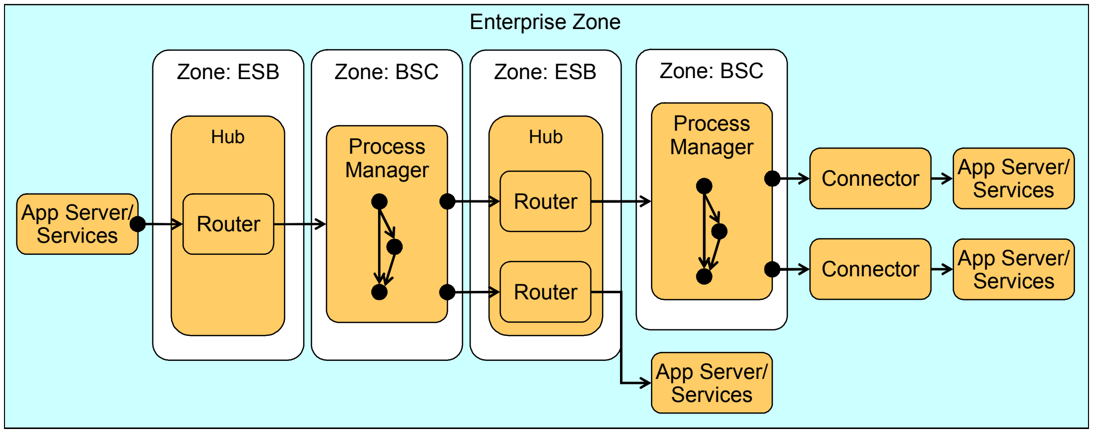


Figure 16 An example of fractal decomposition of choreography

It is worth observing that the Business Service Choreography Zones in Figure 16 can be implemented in a single or multiple business process management engines. With multiple engines, each could be a different instance of the same technology, or two separate technologies could be implemented. Moreover, multiple instances of an engine for each Business Service Choreography Zone can be deployed.

What is important is that the modeling tooling provided is appropriate for the skills of the people modeling the choreography from the requirements. In this example, there are two types of choreography and therefore, two separate modeling tools may be appropriate, each supporting different parts of the spectrum of service choreography, one for each groups of modelers. An example product mapping is shown in Figure 17 on page 29.

The IBM business service choreography capability

We now consider IBM's current technology offerings for business service choreography and how they provide support for BPEL4WS and Web services to interact with an ESB in a service-oriented architecture. First, we look at BPEL4WS.

BPEL4WS

With the backing of organizations including IBM, Microsoft® and BEA, Business Process Execution Language for Web Services (BPEL4WS) is emerging as the leading open standard for describing business processes. It builds upon Web services standards which achieve interoperability between applications by specifying the behavior of business processes between Web services and as Web services. Specifically, activities which make up the business process are encapsulated as services using WSDL and the process itself is exposed as a WSDL definition.

While WSDL enables stateless interaction of Web services, this is not sufficient to support the complex exchanges that characterize business interactions. Support for long-running interactions between service providers, such as business partners, and state management are provided in BPEL4WS.

BPEL4WS provides an industry-wide language for expressing business processes consisting of functions defined through WSDL interfaces, which facilitate portability of business processes across multiple process engines.

The current version of BPEL4WS is V1.1. It is in draft and has the following characteristics:

- At design time, development or modeling tools can use, import or export BPEL4WS to allow business analysts to specify processes and developers to refine them and bind process steps to specific service implementations.
- Runtime choreography and workflow engines can use BPEL4WS to control the execution of processes and invoke the services required to implement them.
- There is a distinction between an abstract process that represents the public behavior of a process in terms of business protocol and an executable business process that defines a complete process that hides private aspects.

  The abstract process definition can be used publicly between organizations to define the interactions between them. The private aspects hide the decision processes that are internal to an organization, but part of the overall process.
- Arbitrarily complex processes can be built using the constructs defined in the standard.
- Support exists for interruptible and non-interruptible processes.
- Data-dependent behavior can be defined.
- Exception conditions can be defined and acted upon.

Further information about the specification can be found at:

`http://www.ibm.com/developerworks/webservices/library/ws-bpel/`

WebSphere Business Integration Server Foundation V5.1

WebSphere Business Integration Server Foundation V5.1 builds on WebSphere Application Server to provide a premier Java™ 2 Enterprise Edition (J2EE) and Web services technology based application platform for deploying enterprise Web services solutions for dynamic e-business on demand™.

It includes process choreography, which provides IBM WebSphere Application Server with the ability to choreograph intra-enterprise and inter-enterprise services into business processes. Those processes are described using the open-standard Business Process Execution Language for Web Services (BPEL4WS). Each activity in the business process is defined as a service using WSDL. The business process in itself is also exposed as a WSDL-defined Web service.

The business processes that are implemented in an enterprise typically require a mixture of human and IT resources and are supported by process choreography in the WebSphere Business Integration Server Foundation. A process is a directed graph that starts with an Input node and ends with an Output node. The process itself is described in WSDL. Its input and output are described as WSDL messages.

A process can contain many activities. An activity can be the invocation of an EJB, a Java class, a service or another process. A process can also be event-driven. For example, it can be paused, waiting for an event, and then resumed when a message arrives.

Process choreography supports processes that can be:

- Long-running (macro-flow) and interruptible (requiring human intervention)
- Short-running (micro-flow) and part of one business transaction

IBM has implemented extensions to BPEL4WS in WebSphere Business Integration Server Foundation. These are as follows:

- Staff support, enabling an activity to be defined to represent a step in a process which is performed by a person. People are assigned at runtime through the execution of a staff query against an enterprise directory using an LDAP plug-in, for example; processes can determine which activities can be performed by certain people.
- Java support, where a snippet of Java code can be invoked as a step within a process. The in-line Java code has access to all BPEL variables for the process instance and can be used to perform activities or test conditions. One usage is to provide data mappings and transformations between variables.

- Quality of service and runtime behavior extensions:
  - Simple process definition versioning using the `validFrom` attribute.
  - Control of the generation of audit entries in activities for the process based upon the value of the `businessRelevance` attribute.
  - On an invoke node:
    - The `transactionalBehavior` attribute can modify transaction boundaries and set explicit checkpointing on transactions.
    - The `continueOnError` attribute allows the process to enter a stopped state after an infrastructure fault.
    - A time-out expression can be defined, particularly on asynchronous calls.

WebSphere InterChange Server V4.2.2

WebSphere InterChange Server is an integration process management broker commonly used to integrate applications. It has a common business object model in which process logic (called collaboration) executes. Both are isolated from the endpoint applications. This facilitates the reuse of process integration logic, enabling execution consistency and simplified ongoing maintenance.

WebSphere InterChange Server provides business object transformation, intelligent routing of messages and a runtime container for business process integration logic. This means that it is easy to manage stateful interactions between multiple disparate integration end points.

WebSphere InterChange Server has a library of pre-built integration processes and business objects. These describe commonly occurring business process functions typically used to integrate packaged applications.

The capabilities of WebSphere InterChange Server should be considered in conjunction with the WebSphere Business Integration Adapter for Web Services (which includes the SOAP Data Handler). Using WebSphere InterChange Server with the WebSphere Business Integration Adapter for Web Services:

- Collaborations can be exposed as Web services.
- Collaborations can consume Web services using service calls.
- Interactions between service consumers and service providers can be choreographed using WebSphere InterChange Server collaborations.
- SOAP messages can be processed using the SOAP Data Handler.

WebSphere InterChange Server provides limited support for BPEL4WS. This is based upon an early version which allows import and export of collaboration templates as BPEL. Files holding the BPEL process definition and the external interface in a WSDL definition are required. A third file contains information about the graphical layout of any activity diagrams.

Collaboration objects execute as Java classes, not as native BPEL.

WebSphere MQ Workflow V3.5

IBM WebSphere MQ Workflow is an IBM product aimed at helping organizations to automate their business processes. WebSphere MQ Workflow is best suited for automating human-centric business processes, since its strength lies in people-based workflows. WebSphere MQ Workflow supports many different staff delegation algorithms and it can also drive system integration via the implementation of one or more User Program Execution Servers (UPES). A UPES activity within a process sends a WebSphere MQ XML formatted message to a user-defined WebSphere MQ queue. The UPES is a custom program that receives this message and performs the request, constructs an XML response that the WebSphere MQ Workflow server will understand and sends the message back to the server. Since a UPES request is delivered by WebSphere MQ, this means that the UPES can run on any of the many operating system platforms on which WebSphere MQ itself runs.

WebSphere MQ Workflow can be used with WebSphere Business Integration Workbench to provide real-time process tracking. Also, real production metrics from the audit trail can be used within WebSphere Business Integration Monitor to analyze an organization's processes. Using these products together provides any customer with the information necessary to achieve continuous process improvement.

WebSphere MQ Workflow is built upon proven IBM technology. The communication layer is built upon WebSphere MQ while the database can use DB2®. The servers can run on many OS platforms, including z/OS®, providing flexibility for an organization's environment. The operational model that can be developed for a WebSphere MQ Workflow implementation can be designed for highly available environments.

At this time, WebSphere MQ Workflow V3.5 does not provide support for BPEL4WS.

The Web Services Toolkit for WebSphere MQ Workflow provides the following capabilities:

- Invoking a Web service from a workflow activity. Thus, a set of Web services interactions can be choreographed within a business processe.
- Publishing a process model as a Web service.

WebSphere Business Integration Modeler V5.1

IBM WebSphere Business Integration Modeler V5.1 is IBM's modeling tool for business analysts and business personnel to:

- Document, analyze, and optimize business processes.
- Provide a faster start to the deployment of the optimized processes.
- Allow an enterprise to assess return on investment.

This development tool is based on and utilizes the Eclipse development platform as an open standard environment for application development and process modeling tools. This modeling tool can capture a business model and generate BPEL4WS and FDL (Flow Definition Language) output.

At a glance, WebSphere Business Integration Modeler V5.1 provides comprehensive, user-friendly business process modeling and collaboration capabilities to graphically design processes across people, partners, and applications. It supports multiple modeling methodologies and industry standards and allows businesses to quickly redesign processes as business needs change. Existing Visio files can be imported as process models and supporting artefacts.

Within an enterprise, this development tool also provides a team environment to share and maintain versions of models, and helps unify in one solution activity costing, quality documentation, business process re-engineering, costing and return-on-investment data. Business analysts can exploit its capability to optimize and project business benefits through “what-if” simulations of models, and also to produce reporting for further analysis.

WebSphere Business Integration Modeler V5.1 can be used to define process models for execution in WebSphere Business Integration Server Foundation V5.1 and WebSphere MQ Workflow V3.5.

- BPEL4WS is generated from process models defined in WebSphere Business Integration Modeler V5.1 for execution in WebSphere Business Integration Server Foundation V5.1.

  The BPEL mode should be used when building process models so that the appropriate validation can be performed. Support is provided for the IBM extensions to BPEL4WS in WebSphere Business Integration Modeler V5.1. BPEL, WSDL and XSD files are then generated.

  These files can be imported into WebSphere Studio Application Developer Integration Edition V5.1. Java snippet code can be added here and then the process can be deployed to WebSphere Business Integration Server Foundation V5.1 for execution.

- FDL is generated from process models defined in WebSphere Business Integration Modeler V5.1 for execution in WebSphere MQ Workflow V3.5. In this case, the FDL mode should be used when building process models.

The modes within WebSphere Business Integration Modeler V5.1 make available different definition and property options on a single common process model. Whether using the BPEL mode to generate BPEL4WS or the FDL mode to generate FDL, other modes, such as the Operational mode, can be used by business analysts to define business process models.

Note: This is a fast moving area of technology. IBM has many initiatives which are focused on providing increasing support for business service choreography. The reader is advised to check for the latest updates for the products described in this section at:

http://www.ibm.com/software/integration/

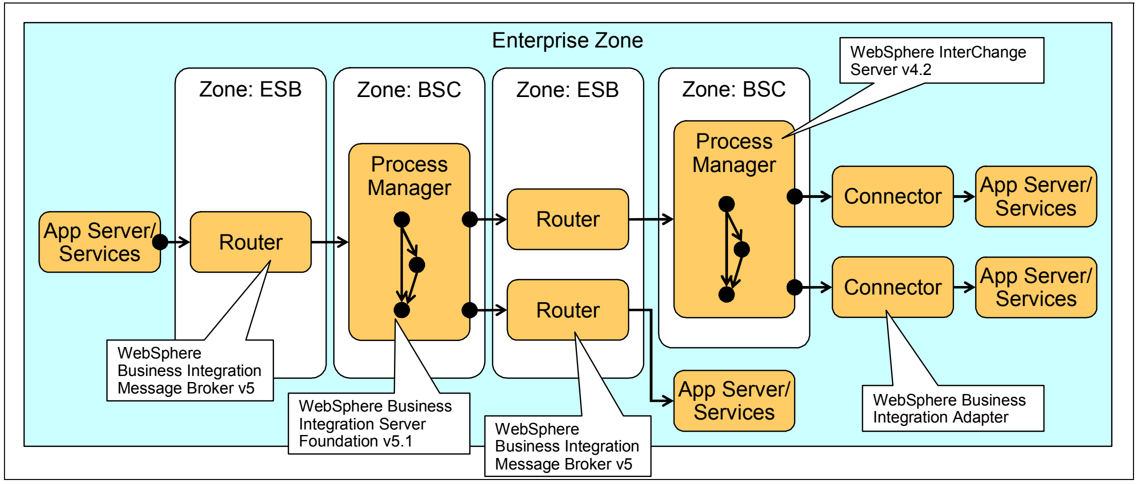
Using IBM business service choreography with an ESB

The combination of these technologies provides IBM with a broad capability to meet the demands across the spectrum of business service choreography. While comprehensive support for BPEL4WS is not present across all of IBM's business service choreography products today, IBM has incorporated the more important standards for interoperability into its products, providing support for Web services, since Web services enable broader architectural interoperability with application components, service consumers and providers and the Enterprise Service Bus.

Example product mappings

The capabilities of the various IBM process offerings can be mapped against the SOA runtime patterns. Standards (such as the Basic Profile defined by the Web Services Interoperability Organization) are a key architectural principle in maximizing interoperability. Existing skills in technologies and products will also provide input for component selection. Based on these factors, there are many selection possibilities.

The fractal decomposition of the Business Service Choreography Zone was shown as a runtime pattern in Figure 16 on page 22. An example component selection for this pattern is shown in Figure 17 on page 29.



Enterprise Zone
Zone: ESB
Zone: BSC
Zone: ESB
Zone: BSC
WebSphere InterChange Server v4.2
App Server/ Services
Router
Process Manager
Router
Router
Process Manager
Connector
Connector
App Server/ Services
App Server/ Services
App Server/ Services
WebSphere Business Integration Message Broker v5
WebSphere Business Integration Server Foundation v5.1
WebSphere Business Integration Message Broker v5
WebSphere Business Integration Adapter

Figure 17   Example product mapping for Figure 16 on page 22

A second example of component selection is shown in Figure 18. This takes the ESB Gateway design pattern from Figure 13 on page 20. Note that there is an intermediate step to derive a runtime pattern.

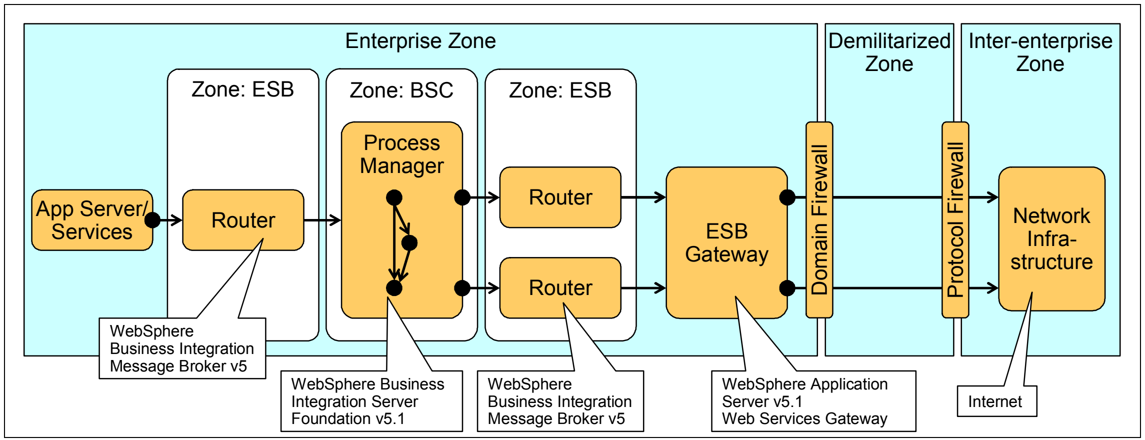


Figure 18   Example product mapping for extended enterprise environment design shown in Figure 13 on page 20

References

1. IBM Redbook, *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346
2. IBM Redbook, *Patterns: Serial and Parallel Processes for Process Choreography and Workflow*, SG24-6306
3. IBM Redbook, *On demand Operating Environment: Creating Business Flexibility*, SG24-6633
4. *Understanding BPEL4WS Parts 1, 2, 3, 6, 8* available on IBM developerWorks® at:

   http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1
5. *Web Services Orchestration: Composition Converges Around Process*, IDC #30222 available from:

   http://www.idc.com/getdoc.jsp?containerId=30222

IBM Redbooks™ are available at:

http://www.ibm.com/redbooks

The IBM Patterns for e-business can be found at:

http://www.ibm.com/developerWorks/patterns

The team that wrote this Redpaper

Chris Nott is a Consulting IT Specialist with the IBM Software Group in the UK. He has fourteen years of IT experience in software development, technical pre-sales consulting and solution architecture. His current area of expertise is business integration and he has a strong background in relational systems design and database technology. He has written extensively on business integration and the Enterprise Service Bus. He holds a BSc degree in Mathematics from the University of Durham and is registered in the UK as a Chartered Engineer.

Thanks to the following people for their contributions to this project:

- Jonathan Adams, IBM Distinguished Engineer
- Sarah Hill, Senior IT Specialist, IBM Software Group
- Martin Keen, Advisory IT Specialist, International Technical Support Organization
- Rick Robinson, Advisory IT Architect, IBM Software Group
- Paul Verschueren, Consulting e-business Architect, IBM Software Group

Notices

This document created or updated on October 8, 2004.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662, P.O. Box 12195
  Research Triangle Park, NC 27709-2195 U.S.A.

Trademarks