IBM Web services
May 2001

Combine business process management technology and business services to implement complex Web services

by Peter Lambros, Marc-Thomas Schmidt and Claudia Zentner

Contents

Overview

This white paper describes the initial use of process management technology in the context of IBM Web services. Discussion focuses on the aspects of business process management (BPM) technology that facilitate interactions between business services, and composition of elemental business services into more complex business services:

- *Workflows support realization of business processes; a workflow process can choreograph the interaction of a set of business services to achieve a business goal. IBM MQSeries® Workflow provides the environment for management of long-running, stateful business processes.*
- *Message brokering facilitates information interchange between business services and provides information routing and transformation services. IBM MQSeries Integrator supports the implementation of message flows that define processing of in-flight information by a set of message processing services.*
- *Micro-flows support realization of complex operations of business services; a WebSphere micro-flow can choreograph the interaction of a set of business services to process the business function offered by an operation.*

You can apply this technology to the context of Web services in a straightforward fashion—a Web service is a special kind of business service. You can use BPM technology to implement complex Web services using intra-enterprise business services or to choreograph interaction of Web services.

This paper describes the extent that IBM supports construction of Web services using BPM technologies and exploitation of Web services in BPM scenarios today.

A simple scenario

The following example illustrates how you can use BPM technology to realize intra-enterprise business processes and the role of Web services in integrating these processes with inter-enterprise activities.

Realizing business services

An example—an enterprise wants to automate the processing of purchase orders. The enterprise defines the business goals it wants to achieve by a *purchase order handling* business service and identifies the business functions (operations) offered by that service (for example, submit, modify or cancel a purchase order). The enterprise then designs the realization of each operation, using existing IT components (for example, enterprise resource planning (ERP) systems, IBM CICS® transactions) or new business services to support specific aspects of the overall process (for example, implemented as Enterprise JavaBeans™ (EJB™).

An example of a *submit purchase order* process:

- *Check the credit of the customer who places an order; initially, this is a service realized by a human process participant who performs the necessary investigation. If the credit check fails, the service rejects the order and another human process participant initiates the necessary exception handling.*
- *If the credit check passes, it uses a prepare delivery business service that does whatever is necessary to make the required goods available for shipment to the customer. The ERP system implements this service; exploitation of these functions may require application of BPM technologies, such as micro-flow-based scripting of business functions or message brokering between the application components involved. This service may require production of out-of-stock goods, that is, it may take a while for this processing step to finish.*
- *When products are ready for shipment, a ship goods business service initiates shipment and handles payment. IBM WebSphere® Business Components implement the service that applies the business logic necessary to perform the required functions, using existing IT components in CICS and IBM DB2®. The service uses micro-flows to describe sequencing of operations compared to the existing IT components necessary to apply the required business functions.*

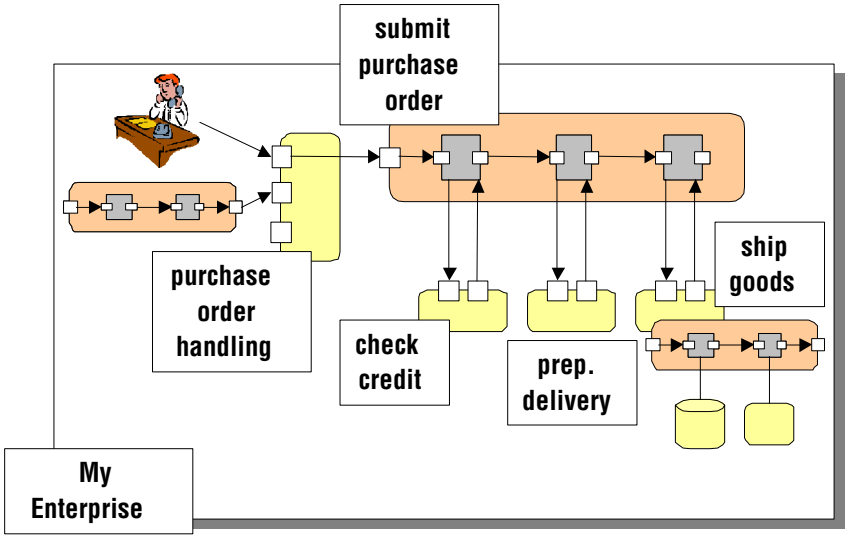

Figure 1. Business service example

Long-running business processes produce workflow processes that choreograph interaction of business services described in Figure 1 that implement the submit purchase order operation. You can define the process using a graphical workflow process modeler and execute it with MQSeries Workflow runtime environment. IBM WebSphere Business Component development tools implements the business services, which include the graphical composition editor for construction of micro-flows and adapter builders that facilitate integration of legacy application components.

The business service implements overall purchase order handling using one or more message flows, executing these services in the MQSeries Integrator runtime environment. These message flows provide the required mediation between the external Web services definition and the internal business service definition, transforming the request and reply messages as necessary and selecting an appropriate target service implementation.

Advertising business services as Web services

In the example, the enterprise designed the purchase order handling business service to support employees who receive purchase orders from customers through fax, phone or e-mail and use the services to process them. The enterprise can make this service available to selected customers and can advertise the purchase order handling business service as a Web service.

The enterprise defines the external interface of the Web service it wants to make available to its trading partners, publishes that interface in a service registry and provides an implementation of the new Web services based on its existing business services. Trading partners can find the Web services definition in the registry and can now directly use operations provided by that service rather than depending on a human intermediary.

Initially, the enterprise publishes a straightforward Web Services definition Definition Language (WSDL) rendering of the interface of the given business service; for example, the Web service has a submit purchase order operation that takes the data necessary to drive the corresponding workflow process as input and delegates processing of the request to that process. This style of Web service, based on a given business service, is useful if all potential users of the service choose to adjust their service requests to the given interface.

This is not always possible—for example, you may want to submit purchase orders in a format different from the one expected by the business service. Therefore, the enterprise decides to support a public standard for processing purchase orders for the second phase of its Web services rollout.

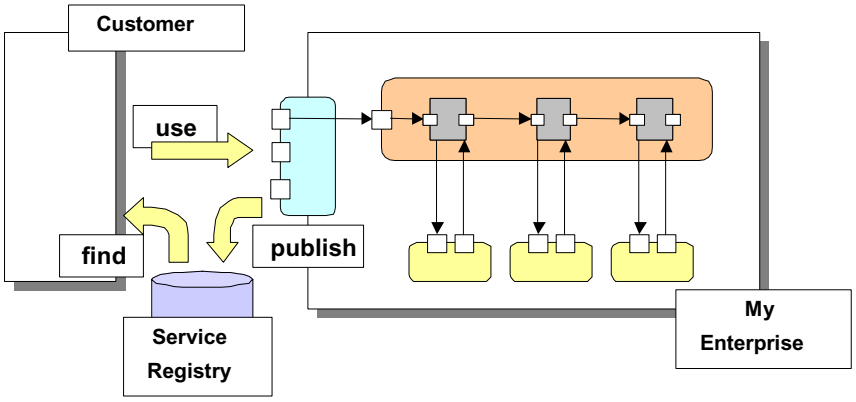

Figure 2. Straightforward publication of a business service as a Web service

In this meet-in-the-middle style of Web service, based on a given Web services interface (defined by the standard) on one side and a given intra-enterprise business service on the other side, a transformation requires a matchmaking between the two components. This is not fundamentally different from other integration problems encountered in BPM applications and BPM technology, such as message brokering among application components or micro-flow-based scripting of business functions. This meet-in-the-middle style is applied to achieve the integration between the WebSphere software application component that implements the Web service and the component that implements the internal business process.

Using Web services as components in business services

Based on positive experiences with Web services, the enterprise can out-source the processing of the credit checks to an external Web service. Enterprises can use a check credit Web service as part of the workflow process described earlier.

Enterprises can find credit checking Web services that match their requirements in a service registry. The enterprise can implement a proxy application component within WebSphere Application Server that supports invocation of the external Web service and modifies the workflow process to use that application component as the realization of the corresponding process step.

Ideally, the enterprise finds a Web service that implements the interface of the intra-enterprise service used before. For example, it may convince a service provider to implement a straightforward WSDL rendering of the interface of its internal credit checking business service in a top-down style of Web service realization. In this case, the interaction with a Web service replaces in the workflow process.

Alternatively, the company can use the workflow process editor to modify the process to use the new service as the credit-checking step and redeploys the workflow process. In addition, use other BPM technologies, such as message brokering and information transcoding, in a micro-flow to mediate among the requirements of the business process and the services offered by the external service provider.

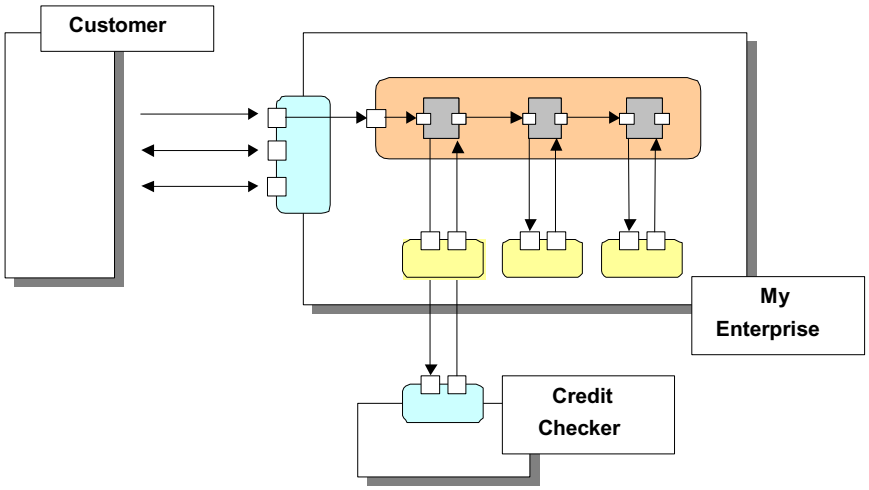

Figure 3. Web services as components in business services

Web services and BPM technology

Web services are sets of business operations that an enterprise makes available through the Internet. You deploy Web services on network-accessible platforms and describe Web services with a service description language, the WSDL. The service description contains the interface of the service, its data types, binding information and its network locations. The service description is published to a registry. The three main roles in Web services architecture are:

- *The service provider—from a business perspective, this is the owner of the service. From an architectural perspective, this platform provides access to the service.*
- *The service requester—from a business perspective, this business requires certain function to be satisfied. From an architectural perspective, this application is looking for and invoking a service.*
- *The service registry—a searchable repository of service descriptions where service providers publish their services descriptions and service requesters find and obtain binding information for services. Universal Description, Discovery and Integration (UDDI) is an example of a service registry.*

Web services tooling

The Web Services Toolkit and the eXtensible markup lanuguage (XML) and Web Services Development Environment, available from IBM alphaWorks® support the integration between intra-enterprise components and Web services. They offer tools to help perform the following tasks:

- *Publish, unpublish and find Web services that are defined using WSDL, and registered and stored in a UDDI registry*
- *Create WSDL documents from existing Java™ classes, servlets and EJB*
- *Generate client and server code from a WSDL document*
- *Deploy Web services to a WebSphere Application Server*

In addition, they provide tools for building DTDs, XML Schemas, XSLT, mappings between XML and different back-end stores. This Web services tooling supports the following development scenarios:

- *Top-down—for a given WSDL definition of a Web services interface, the Web services tooling generates a (Java) client proxy, which uses service requesters to interact with a remote Web service implementing the WSDL interface. It also generates a server-side (Java) skeleton that gives the service provider a starting point for the implementation of the actual Web service and supports the deployment of this implementation.*

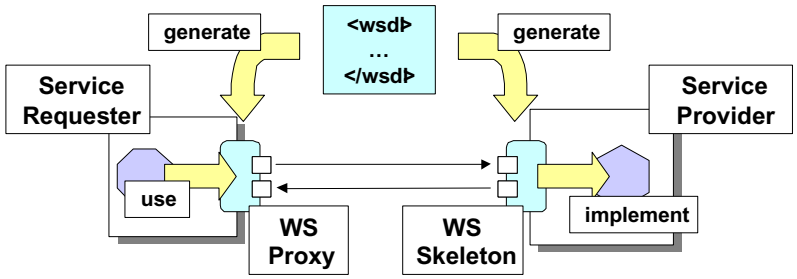

Figure 4. Top-down Web services development

- *Bottom-up—for a given Java component, the Web services tooling generates a WSDL definition that exposes the operations of the Java component as a Web service and supports deployment of the Java component as the realization of the service. This includes generation of the necessary deployment descriptor for the Web service, which defines the association between Web service requests and operations on the actual service implementation. As in the top-down scenario, the WSDL generates a proxy to support client access to the Web service.*

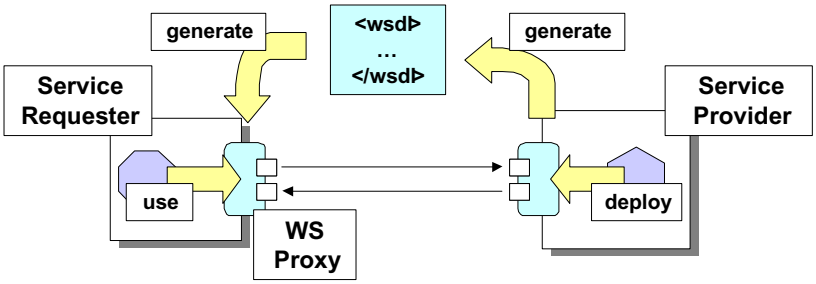

Figure 5. Bottom-up Web service development

You can use Web services tooling in a straightforward fashion to advertise intra-enterprise software components as Web services, to facilitate realization of Web services by intra-enterprise components and to access external Web services from intra-enterprise applications.

Business process management technology

The following sections provide an overview on business process management technologies that facilitate interaction of a broad spectrum of complexities of business services and choreograph these services to realize business processes.

Flow composition

The general theme for the following technologies is flow composition. Each uses the notion of a flow model in one fashion or another. A flow model is a directed graph with nodes representing usage of business services and arcs representing the flow of information and control between these services.

The flow composition model (FCM) defines the general specification of these flow models. Modeling tools and associated runtime environments support this model. You construct flow models from a palette of flow components, which represent business services that can contribute to the realization of a complex business service.

Strong similarity exists among the three variations on flow modeling in the following discussions (work- message- and micro-flows). However, they all address different aspects of the overall business process management space, and as a result, the runtime behavior of the resulting flow models is different-just as the kind of flow components that they use. The following sections discuss each in more detail.

Business processes and workflows

MQSeries workflow (MQWF) supports specification and execution of long-running, interruptible, potentially interactive business processes. A workflow process comprises a set of activities. Each activity represents a business service that contributes to the overall business goal of the process. A single operation provided by some software element component or by a sub-flow can implement an activity. You can also define an activity to interact with people, for example, to handle exceptions. Connectors between activities determine control and data flow within a process; business rules can be associated to connectors to control the paths through a process, as well as to data mapping to specify how you can pass data between activities.

The MQWF process-modeling tool supports definition of workflow models and related meta-data (for example, data structures for activity signatures and program definitions for activity implementations). Workflow models are translated into an external process specification language (FDL) that is deployed to the MQWF process execution environment. The MQWF process engine interprets these models and invokes the software components that implement activities in the workflow model.

Business services and micro-flows

Micro-flows are used in the WebSphere Business Integrator and the MQSeries Adapter Offering use micro-flows to describe the behavior of a particular operation of a business service. They describe the steps necessary to perform the operation and the control and data flow between these processing steps. For each processing step, you can define a flow component that performs the actual business function required for that process step. In many cases, a flow component represents the invocation of an operation of another business service (for example, an EJB or an adapter to some existing application), but components can also provide the necessary data mapping between processing steps, and represent routing decisions in the control flow.

Existing Java components can import into the micro-flow modeling tool and are made available on the flow composition palette. The modeling tool supports generation of executables that implement the business logic defined in micro-flows and invoke the Java components that realize flow components in the micro-flow model. The generator translates a set of micro-flows into a Session EJB with one operation per micro-flow.

In contrast with workflows, micro-flows are short-lived and have no persistent state; they execute in the transaction context of their host. For example, the host can be a Session EJB that uses micro-flows as operation realizations.

Intelligent information bus and message flows

MQSeries Integrator (MQSI) supports specification and execution of message flows that define rules for in-flight information processing. A message flow describes the flow of information and messages between a set of message-processing nodes; it is short-lived and carries no persistent state. Message-processing nodes take a message as input and produce one or more messages as output. MQSI provides a set of processing nodes to support message transformation and augmentation, eventually exploiting information stored in some relational database management system (RDBMS). It also allows for realization of additional, user-defined message processing nodes. Message flows deploy in the MQSI execution environment. A business event (a message arriving on a queue) triggers a message flow.

Message flows can act as intermediaries that sit between requesters and providers of business services. In this capacity the message flow can transform (syntactically or semantically) a service request to adapt a required interface to a provider's published interface, thus giving a nonintrusive mechanism for integrating a wide range of service or application providers.

In addition, a message flow represents a natural paradigm for providing transparent routing (perhaps based on class of service or services), such as auditing, warehousing or nonrepudiation. When combined with the integrated publish and subscribe or event management capabilities of the message flow infrastructure, this allows clients and providers to establish and manage negotiations.

Combining BPM technology and Web services

Application of the business process management (BPM) technologies to Web services is straightforward. You can substitute the term Web service virtually anywhere in place of the term, business service.

The following can help you distinguishes two application scenarios: one uses BPM technologies to realize complex Web services; the other one uses Web services as components in complex integration scenarios.

Web services and MQSeries workflow

The relationship between an MQWF and Web services is twofold: an MQWF process can provide implementation for a Web service, and an MQWF process can include Web services as implementation for its activities.

Using a process as implementation for a Web service allows composing complex Web services with the characteristics of a process as explained earlier. The actual parameters of the Web service will determine the actual business context of the process. And the process can base on other Web services, such as its implementation, as well as other business tasks (for example, staff-related activities).

In short, this means:

- *Allow to execute an MQWF process as Web service*
- *Allow to invoke a Web service as an activity's implementation*

With its message-based XML interface, MQWF allows to execute process instances, and a process instance to invoke activity implementations by means of sending and receiving XML messages.

The following sections explain how MQWF can be a component in Web services architecture by exploiting this XML interface today.

Processes as Web services

Flow Definition Language (FDL) defines an MQWF process. From this process definition, you can derive the definition of a Web service that is implemented by this process, including the Simple Access Object Protocol (SOAP) binding for this Web service. The following figures show an FDL item describing the interface of a simple workflow process and the WSDL rendering of this interface.

```
STRUCTURE 'CreditInfo'
 'CreditRequestor': STRING;
 'CreditAmount': LONG;
 ...
END 'CreditInfo'

PROCESS 'CreditRequest' ( 'CreditInfo', 'CreditInfo' )
 ...
END 'CreditRequest'
```

Figure 6. Sample process definition (FDL

```
<?xml version='1.0'?>
<definitions name="MQWFService" … >
  <message name="CreditInfoInput"> <part name="CreditInfo" type="xsd1:CreditInfo"/>
  </message>
  <message name="CreditInfoOutput"> <part name="CreditInfo" type="xsd1:CreditInfo"/>
  </message>
  <portType name="CreditRequest">
   <operation name="CreditRequest">
     <input message="CreditInfoInput"/>
     <output message="CreditInfoOutput"/>
   </operation>
  </portType>
  <binding name="CreditRequestSOAPBinding" type="CreditRequest"> … </binding>
  <service name="CreditAgencyService">
   <port name="CreditRequestPort" binding="CreditRequestSOAPBinding">
     <soap:address location="http://…"/>
   </port>
  </service>
  <types>
   <xsd:schema … >
    <xsd:complexType name='CreditInfo'>
     <xsd:element name='CreditRequestor' type='xsd:string' minOccurs='0'
                  maxOccurs='1'/>
     <xsd:element name='CreditAmount' type='xsd:integer' minOccurs='0'
                  maxOccurs='1'/>
    </xsd:complexType>
   </xsd:schema>
  </types>
</definitions>
```

Figure 7. Corresponding WSDL

A Web service request submitted by a service requester is intercepted and transformed into a format understood by an MQWF. Consequently, the implementation of the service acts as a service adapter; it handles the transformation from the service request to the XML message that is understood by MQWFs runtime to execute a process instance, and from the MQWF XML response message to the service result.

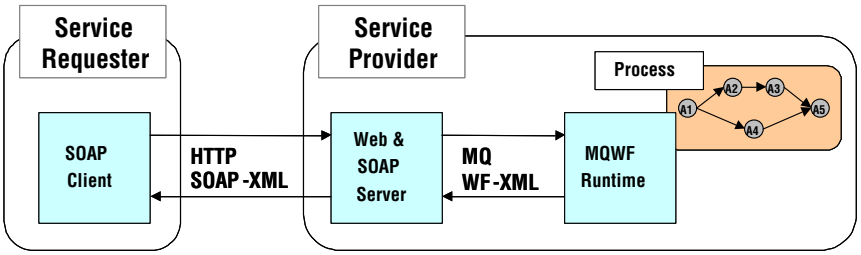

Figure 8. Invoke process as Web service (sample scenario)

Sample SOAP request → **Sample MQWF XML request**

```
<SOAP-ENV:Envelope
 xmlns :SOAP-ENV="http://schemasxmlsoap.org/soap/envelope/"
 SOAP-ENV: encodingStyle =
 "http://schemas.  xmlsoap .org/soap/encoding/">
   <SOAP-ENV:Body>
     <m: CreditRequest xmlns  :m="Some -URI">
       <CreditInfo >
         <CreditRequestor  >A</ CreditRequestor  >
         <CreditAmount  >1000</ CreditAmount  >
       </CreditInfo >
     </m: CreditRequest  >
   </SOAP -ENV:Body>
</SOAP -ENV:Envelope>
```

```
<WfMessage >
 <ProcessTemplateExecute   >
  <ProcTemplName  >
     CreditRequest
  </ ProcTemplName  >
  <ProcInstInputData  >
    <CreditInfo >
      <CreditRequestor  >A</ CreditRequestor  >
      <CreditAmount  >1000</ CreditAmount  >
    </CreditInfo >
  </ ProcInstInputData  >
 </ ProcessTemplateExecute   >
</ WfMessage >
```

Figure 9 Corresponding sample SOAP and MQWF XML request messages

You need to take the following steps to make a given workflow process available as realization of a Web service:

- *The service provider models a process using the MQWF process modeler and generates the FDL representation of this process.*
- *The service provider generates a WSDL file from this FDL by means of a converter that extracts the process interface definitions from FDL. This WSDL serves as the Web service interface description for service requesters.*
- *The WSDL file can be imported into the Web service tooling. The tooling helps to perform the following steps:*
  - *Generation of a server-side skeleton Java Beans implementation from WSDL to be used by the service provider. This Java bean implements the service. Its implementation sends an XML message to MQWF that corresponds to the service to invoke the process. It can receive an XML response message from MQWF and transform it into the result of the service.*
  - *Deployment of this Java bean as Web service to the service provider's WebSphere Application Server. As soon as the Web service is deployed, it is ready to be invoked by service requesters.*
  - *Generation of a Java client proxy from WSDL, which can be used by the service requester to access the Web service.*

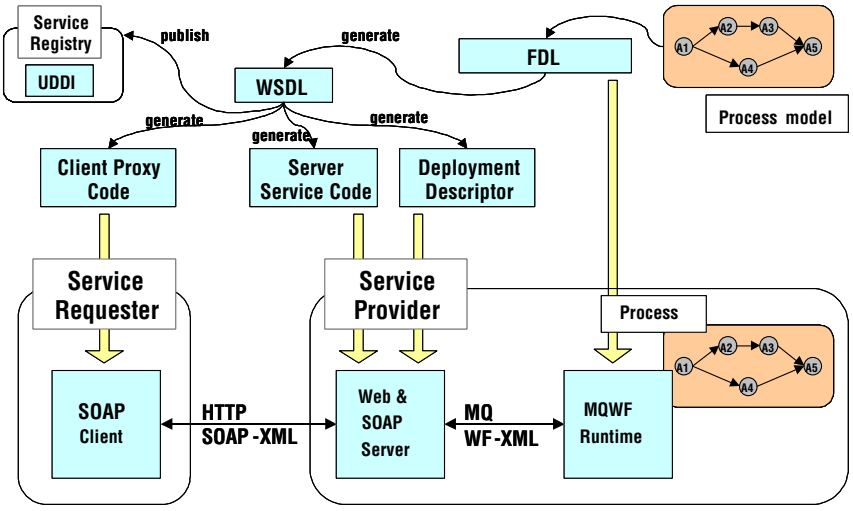

Figure 10. Processes as Web services

Web services as activity implementations

A Web service can provide the implementation for an activity within the context of an MQWF process; a process thus acts as service requester. A user-defined program execution server (UPES) is the component to handle the invocation of a Web service as implementation of an activity. When the activity implementation invokes as part of the process navigation, MQWF sends an XML message to the UPES. This message carries all the data needed for the service invocation; the content of the message identifies the Web service and specifies its parameters. Acting as a client to the Web service, the UPES calls the specified Web service through a proxy. Optionally the UPES can return the result of the Web service as an XML response message to MQWF. This message signals the completion of the activity's implementation.

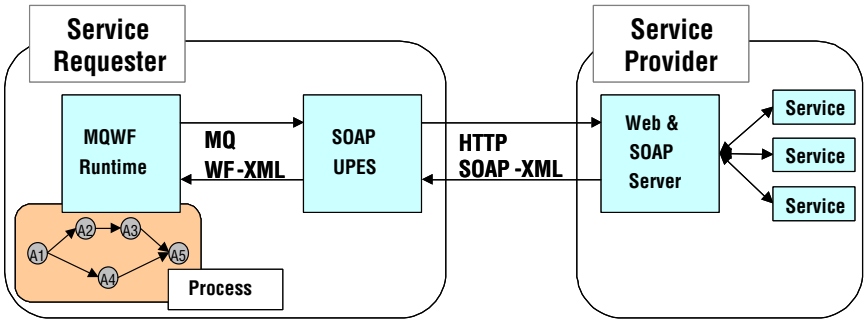

Figure 11. Invoke Web service as activity implementation (sample scenario)

You need to take the following steps to use a given WSDL-defined Web service as an implementation of an activity in an MQWF-driven process:

- *The service requester models an MQWF process. If a given Web service uses as step within the process, the specification of the WSDL determines the specification of the activity that implements this service. Definitions for MQWF modeling constructs are derived from the service specification: the service signature determines the activity's signature; input and output messages of the service determine data structure definitions; a program definition specifies the Web service invocation details.*
- *The UPES of the service requester supports the invocation of this Web service. In case of a UPES implementation that handles service invocations generically, the client proxy of the Web service is to be deployed to this infrastructure; the Web service tooling allows generating a Java client proxy from WSDL. The UPES transforms an incoming MQWF XML message into a call of the corresponding Web service as well as it transforms the result of the Web service that is to be sent back to MQWF in the XML format expected by MQWF.*

Static specification of a Web service as implementation for an activity during design time is an option. Dynamic lookup and binding of a Web service just-in-time during the execution of a process is an advanced option. This requires interaction with a service registry during runtime to find and bind a Web service based on some criteria that was specified during design time.

Web services and MQSeries Integrator

One of the primary roles that MQSI can play in the context of Web services is that of the service mediator. As would be expected we have the usual twofold relationship; first, a message flow advertises and implements a Web services, and second, Web services invoke as a processing step or node within a message flow.

An MQSI message flow is may be constructed using the WSDL definitions that are generated by the Web services tooling. Message processing nodes are added to the flow to provide the desired mediation capabilities. For example, on receiving a Web service request, a flow may transform this request message

into an alternate Web service request (for example, it may alter the routing information or it may modify the actual content of the message in some way) or it may capture data from the request for in-flight logging in a warehouse or audit system.

Furthermore, because a message flow is acting as an intermediary rather than directly as a service implementation, this paper does not address the need to advertise a generic message flow as a Web service at this time but rather observes that message flows can be constructed to handle a predefined set of Web services. The following sections discuss this in detail.

Message flows as Web services

Message flows are used to implement operations of Web services that require mediation between the originating request and the service providers or, require delegation to a set of service providers is required and their answers are aggregated. Another option is to wait for the fastest response of one provider. At a basic level, this may mean transforming the content and/or representation of a request from the format expected by the client to the format offered by the available service providers. In this way, the mediation or transformation separates from the core business implementation and is more adaptable as the needs of the business change or as new providers become available. A message flow can also provide sophistication in value-add services, such as audit, warehousing, content-based access control, nonrepudiation, intelligent routing or time-of-day-based processing and deliver these sophistications in a flexible and nonintrusive way.

Message flows are also used to implement dynamic service aliasing, based on a variety of *quality-of-service* considerations or to provide dynamic lookup for internal rather than external Web service names. The message flow paradigm works well to provide dynamic routing and mapping, based on the context and content of the message, which carries the data.

You can consider a bottom-up scenario first. An arbitrary message flow is not required to support a request and reply model. In general, you can expect to need a pair of flows (or at least a complex flow with two execution paths) to implement a Web service. One flow mediates the inbound request and the other flow mediates the outbound response. However, a single flow can perform a synchronous set of operations and respond directly to the initiator of the flow (and the product includes a *Reply* node in support of this model). A particular example of this is the execution of a simple stored procedure, which provides the underlying implementation of the requested service.

A message flow invokes by using a defined set of (one or more) message formats but the current implementation of MQSI does not provide direct support for SOAP requests over HTTP. You can use a proxy Web service to provide the bridge into the MQSI environment. The simplest solution passes the body of the SOAP request directly over MQSeries into MQSI but in general, this requires an additional transformation at the start of the flow and to handle the response from the flow.

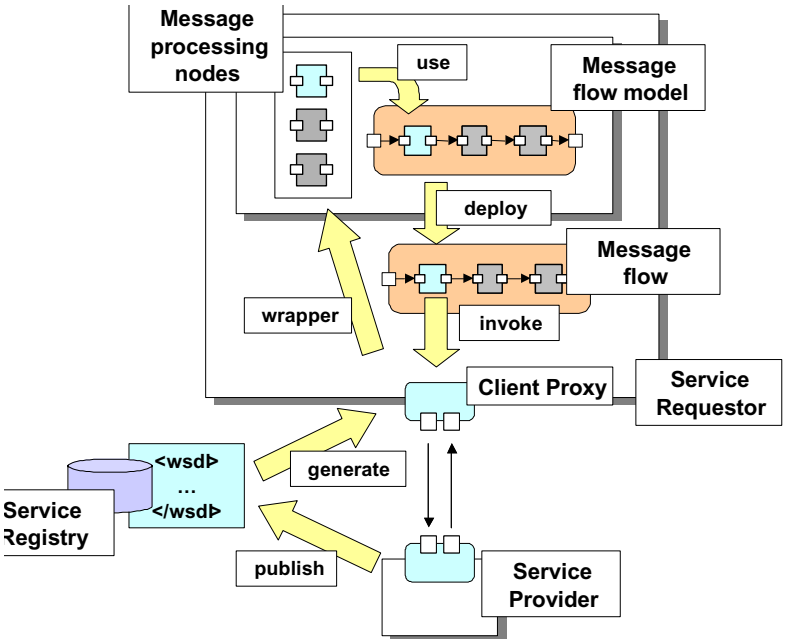

Figure 12. Message flows as Web services

You can build these additional transformations using the standard MQSI tools to map the request and response into the form required by the particular implementation of the message flow. This provides complete flexibility in being able to map the Web service request and response to any other message format, which is supported by the MQSI broker.

In a top-down scenario, you can use a given WSDL operation definition to develop a message flow, which consumes and produces the required input and output message definitions. Add message definitions for the request and response flows to the MQSI message repository and use the message flow builder tool to construct the desired message flow by adding and customizing processing nodes for transformation, enrichment, data capture or routing. For example, the message flow can use an MQSI compute node to specify one out of a number of possible providers to which to route the request based on the identity of the requestor or the actual content of the request. Alternatively, you can extract data from the request and captured into a relational database for offline traffic analysis by inserting an MQSI Data Insert node into the message flow.

The asynchronous nature of MQSI, especially when used in conjunction with the publish and subscribe functions of the product, also lends itself to supporting the push models of Web services. Service requestors can register an interest in certain sets of services; this can be mapped by a proxy service into an MQSI subscription request. Service providers can publish information, which will be processed by the MQSI matching engine and distributed as pushed messages to the appropriate set of clients. All mappings between the Web services domain and the MQSI domain are handled by a proxy service.

Web services as message-flow components

Message flows can use Web services as realizations of message-processing nodes in a message flow. As explained above, message-flows primarily perform message manipulation and routing functions and Web services can be used in this context, for example, to retrieve information that is added to a message or used to make routing decisions.

A new SOAP node for MQSI can be instantiated within a message flow, providing the capability for a message flow to make a synchronous request to a Web service. This message-processing node builds on the facilities provided by the Web services toolkit to make the request to the desired Web service, receive its response and pass the response data back to the message flow.

The message that initiates the execution of the message flow may already be a SOAP message of the correct format, in which case it passes directly to the SOAP node to invoke the required service. Alternatively, the message flow may be initiated by a message, which cannot be used to invoke the Web service directly; in this case, the message flow uses one of the standard transformation nodes in MQSI to build the appropriate SOAP request.

No matter how the service definition made it to the message-flow builder palette, after it is on the palette, you can construct new message flow models. The signature of this node is such that it accepts a SOAP message object structure as input and produces the same as output.

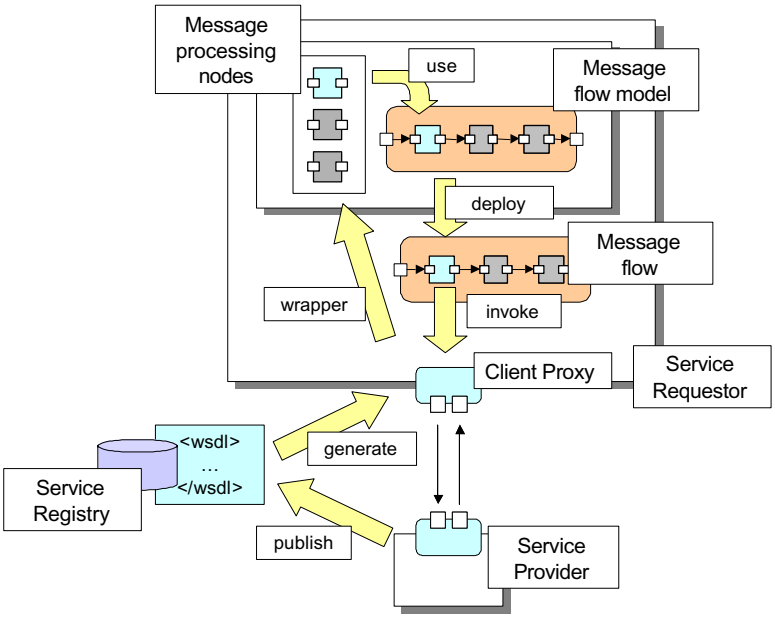

Figure 13. Web services as message flow components

In a bottom-up scenario, a message-flow uses a message-processing node definition to derive requirements for a matching Web service and to generate the WSDL definition of such a service. The signature of the processing node defines the interface of a matching operation of the Web service. In practice, given the nature and scope of currently implemented message-processing nodes, this appears to be a less likely scenario.

Web services and WebSphere micro-flows

Use micro-flows to realize operations on business services. From an implementation perspective, you realize the business service through a stateless EJB session. The Java code generated from micro-flows implements the operations of that EJB. Java components provide realizations of flow components in a micro-flow model.

The Web services tooling provides the necessary means to enable transformation of micro-flows into Web service realizations and invocation of WSDL-defined Web services as nodes in a micro-flow.

Web services Tooling technology can be used to generate a WSDL definition for the Session EJB that aggregates micro-flows and to register that Session EJB as the realization of the resulting Web service. It also supports generation of Java client proxies for given WSDL-defined Web services; these Java components can then be made available on the palette of the micro-flow builder and can be invoked by the micro-flow runtime. The following sections discuss this in detail.

Micro-flows as Web services

Micro-flows can be used to implement operations of Web services that require co-ordination of a set of business components. The micro-flow would describe the sequence of actions to be taken to perform the business function provided by the operation. It performs necessary transformations of input data to fit the requirements of the business components involved in executing the request; it would also aggregate results produced by the various contributors into a result passed back to the service requester. The figure below describes the steps to be taken to use a micro-flow as part of the realization of a Web service:

- *A given micro-flow model compiles into a micro-flow executable using the micro-flow builder. The resulting Java component (a Java bean representing the micro-flow proper or a Session EJB that uses micro-flows as operation implementations) provides the base for generation of the artifacts necessary to make this business component available as a Web service.*
- *The Web services tooling is used to register the micro-flow Java element component with the service invocation mechanism in WebSphere Application Server; service requests received by the SOAP router pass on for processing to the associated micro-flow. In addition, the toolkit would generate the WSDL rendering of the interface of the Java element component that uses the micro-flow as part of its implementation.*
- *As described above, the resulting WSDL definition can be published in some service registry and can be used by service requesters to find and invoke the new Web service.*

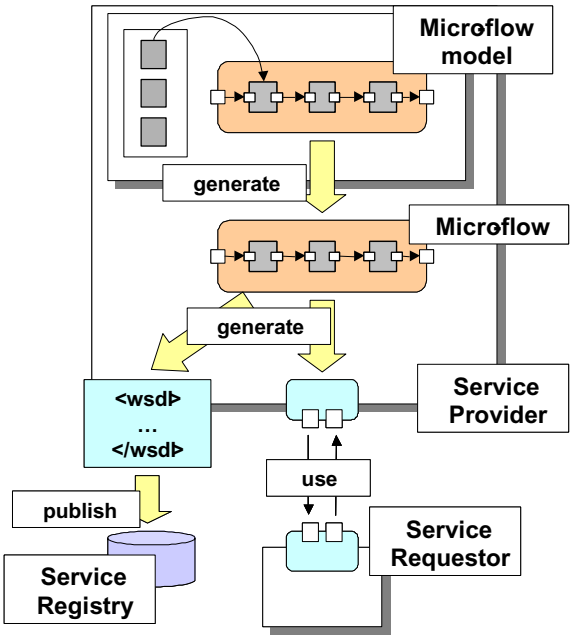

Figure 14. Micro-flows as Web services

In a top-down scenario, a given WSDL operation definition is used to generate a template for a micro-flow that implements the operation. Input respectively output and eventual fault elements of the operation can be translated into input respectively output terminals of the micro-flow with signatures matching the signature of the operation elements.

Web services as Micro-flow components

Micro-flows can use Web services as realizations of flow components.
The figure illustrates the steps necessary to construct a micro-flow that uses a given Web service:

- *For a given WSDL service definition, the Web services tooling generates a Java client proxy that is used by service requesters to interact with the given Web service.*
- *As any Java component, this proxy is added to the palette of flow components in the micro-flow builder. Only a subset of the methods provided by the proxy is exposed on the interface of the resulting flow component.*

- *The new flow component representing the external Web service is used to compose a new micro-flow.*
- *The resulting micro-flow model is translated into an executable micro-flow (for example, an operation on a Session EJB); the Java client proxy for the Web service becomes a flow component implementation for this micro-flow.*
- *When the micro-flow is invoked, it uses the Java client proxy to interact with the external Web service.*

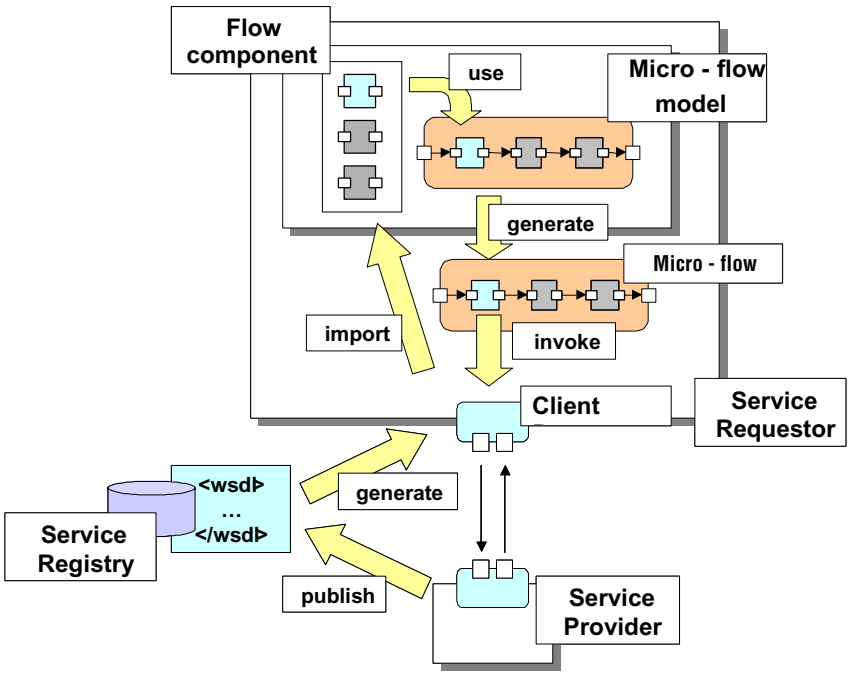

Figure 15. Web services as micro-flow components

In a bottom-up scenario, a flow component definition used in a micro-flow can be used to derive requirements for a matching Web service and to generate the WSDL definition of such a service. The signature of the flow component defines the interface of a matching operation of the Web service.

IBM Corporation
Route 100, Building 1
Somers, NY 10589
U.S.A.

Producdc in the United States of America
05-01

References

1. E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, “Web Services Description Language (WSDL) version 1.1” http://www.w3.org/TR/wsdl March 2001
2. Extensible Markup Language (XML) 1.0 (Second Edition): http://www.w3.org/TR/REC-xml.
3. Simple Object Access Protocol (SOAP) 1.1: http://www.w3.org./TR/2000/NOTE-SOAP-20000508/
4. MQSeries Family: http://www.software.ibm.com/mqseries.
5. The IBM XML and Web services development environment: http://www.alphaworks.ibm.com/tech/wsde.
6. The Web services toolkit: http://www.alphaworks.ibm.com/tech/webservicestoolkit.
7. http://www,ibm.com/developerworks/webservices
8. http://www.uddi.org