



IBM Software Group

# WebSphere® Message Broker Version 6.1

## *Web services SOAP nodes*



@business on demand.

© 2008 IBM Corporation  
Updated January 23, 2008

This presentation will describe the new Web services SOAP nodes, delivered in WebSphere Message Broker Version 6.1. There are two other related sessions, covering WS-Addressing, and the processing of messages in the SOAP domain.

## Agenda

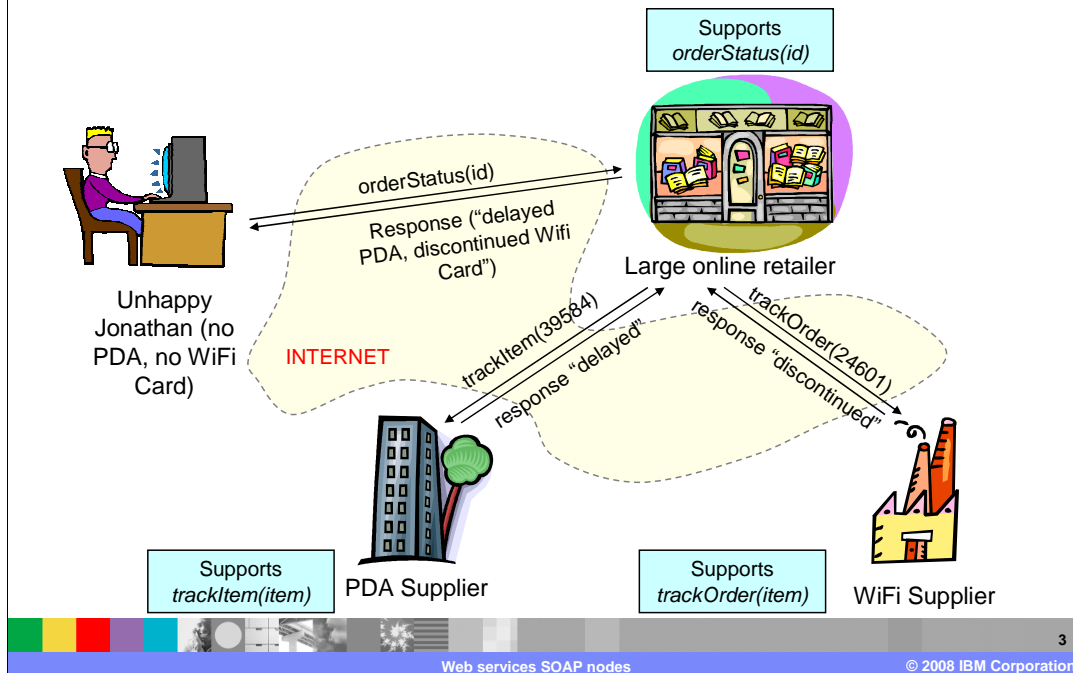
- Overview of Web services
  - ▶ Basic scenario
  - ▶ Terminology
  - ▶ Relating scenarios to message flows and nodes
- New SOAP nodes
  - ▶ SOAP input and SOAP reply nodes
  - ▶ SOAP request node
  - ▶ SOAP AsyncRequest and SOAP AsyncResponse nodes

This session first provides an overview of Web services, and provides a refresher on the basic Web services terminology. It then provides some scenarios that can be related to an implementation within WebSphere Message Broker.

It follows this with a full description of the new SOAP nodes in Message Broker Version 6.1, covering each of the new nodes in some detail.

There are additional sessions, covering the processing of SOAP messages, and the implementation of Web services WS-Addressing, and the use of WS-Security.

## Web services example – Where's my stuff?



This example shows Jonathan, who has not received his WiFi card or PDA from a large online retailer. He wants to query his order, and in this case the retailer provides such a service by exposing a Web service over the internet.

So, Jonathan can send an enquiry to the retailer over the Internet, to see what the status of his order is.

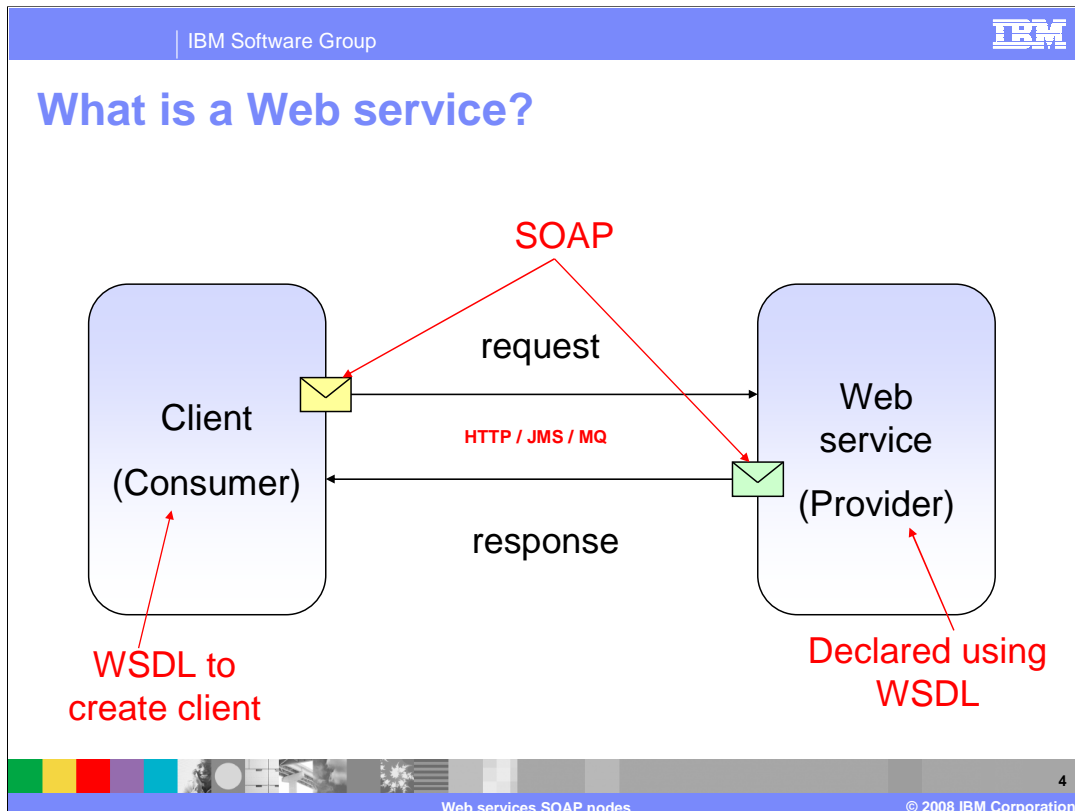
In this case, the retailer has no local stock, so cannot provide a status locally.

However, the PDA Supplier and WiFi Supplier both expose a Web service that can be used by the online retailer.

The online retailer can send a 'request' using Web services, requesting the status of the goods.

Response can then be returned to the retailer with the current status of the products.

The online retailer can then send this information back to Jonathan.



This slide highlights what was taking place on the previous slide.

Each of the requests has a client, or consumer, and a Web service, or provider. For example, Jonathan is a consumer, and the online retailer is a provider.

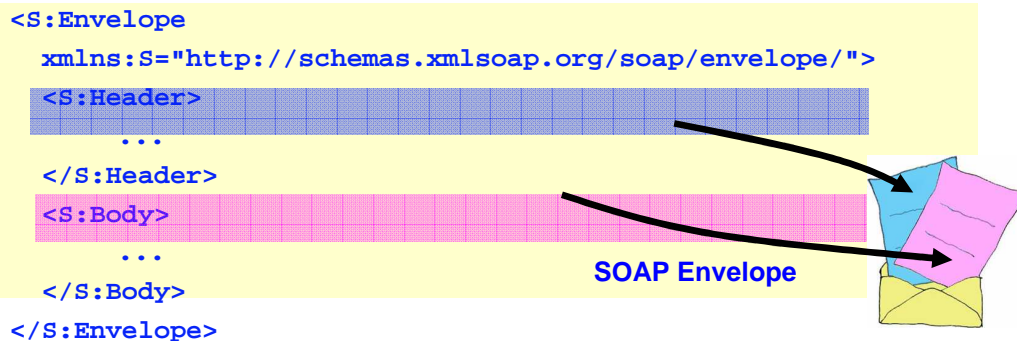
The request from client to Web service is achieved by sending a SOAP message from the consumer to the provider. This message can be sent using any of the three transport protocols, HTTP, JMS or MQ. The Web service provider receives the request, and provides an answer by responding with another SOAP message.

The nature of the request is defined by the Web service provider, and uses Web services definition language, commonly known as "WSDL". The WSDL created by the service provider is published by the provider, and can be used by the consumer to create a request in a way that can be understood by the provider.

The WSDL is also used to describe the type of data structure that is passed from consumer to provider. For example, is it a SOAP message, or a SOAP message with an attachment.

## SOAP

- SOAP is an XML based language defined by the W3C for sending data between applications
- SOAP is transport and platform neutral



SOAP is the standard way of describing messages that are sent between consumers and providers. It uses XML as its basis, and has a very well-defined structure, defined by the W3C organization.

Although this used to stand for “simple object access protocol”, this has recently been dropped, and these messages are now known as “SOAP”. SOAP messages are independent of both the transport software, and the operating system hosting the consumer and provider. The SOAP message contains an envelope, which in turn contains several components.

The first item is the namespace, which contains details about the version of the SOAP message. The example on this slide shows that it is a SOAP Version 1.1. message.

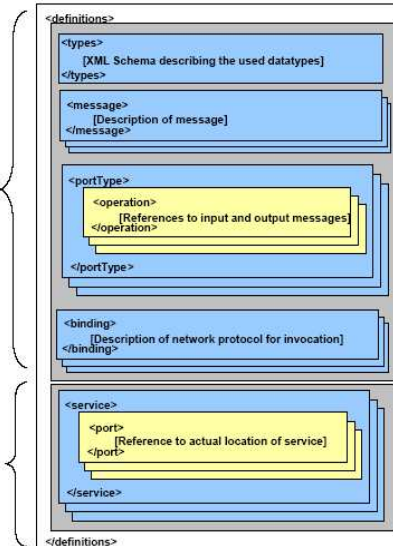
The second item is the SOAP header, which is optional. This can be used for holding information about the message. It can also be used for information about WS-Addressing and WS-Security, two Web services standards which will be discussed in other sessions.

The third item is the SOAP body, which is mandatory. The body is where user data is placed, this is sometimes known as the payload of the message.

It is also used for handling error situations, known as “SOAP faults”. For example, if a consumer sends a request to a provider, but the provider cannot interpret the request, then this error is handled as a SOAP fault. The provider will introduce a SOAP fault part of the message.

## WSDL (“Web services definition language”)

- WSDL is used to define Web services and how to access them
  - Specifies its location
  - Describes the operations (methods) it exposes
- Normally generated automatically
- Each <message> part is of a certain data <type>
  - Could be simple type
  - Could be complex type defined with an XML schema **Interface**
- An <operation> is defined by its input and output <messages>
  - For example, one way, request / response
- A collection of <operations> form a <portType>
  - Defines a set of operations and their associated messages
- A <binding> specifies how operations are accessed using a particular protocol, (for example, SOAP - document literal, RPC)
- A <service> defines the concrete parts of the Web service
  - <port> specifies an address for a <binding>



6

This slide introduces the key concepts in Web services definition language, or WSDL.

WSDL is used to describe everything about a Web service, including its location, and how to invoke it. Different operations can be provided by the Web service, and these are described by the WSDL.

WSDL is normally generated automatically by the application development tools that are used to build the Web service. This can then be used by the service consumer, by importing the generated WSDL into the consumers application development tools.

A WSDL example will have these key parts:

First, a message part, which will have a certain data type. Data types can be simple, for example “string” or “boolean”, or a more complex type defined by an XML schema.

Second, an operation is defined by its input and output messages. It can therefore be a one-way message, which is a simple request, or a two-way message, which is a request-response message.

A collection of these operations is known as a “port-type”.

The “binding” specifies how the operations are accessed, using a particular protocol.

And finally, the “service” definition describes how a consumer can access the provided, by specifying for example the URL of the Web service.

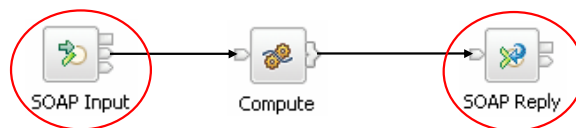
## Section

# *The new SOAP nodes*

This section introduces the new SOAP nodes in Message Broker Version 6.1.

## SOAP input and SOAP reply nodes

- SOAP input and SOAP reply nodes
  - ▶ Construct a message flow which implements a Web service provider
  - ▶ SOAP input node listens for incoming Web service requests
  - ▶ SOAP reply node sends responses back to the client
  - ▶ Between these nodes, normal flow processing logic occurs



First, the SOAP input and SOAP reply nodes.

This slide shows how the SOAP input and SOAP reply nodes will be used in combination. The SOAP input node allows a message flow to be represented as a Web service, and the properties of the SOAP node describe how the Web service request will be delivered and handled by the broker.

The SOAP Reply node sends the response back to the original requester of the service.



## SOAP request node

- SOAP request node
  - ▶ Used in a message flow to call a Web service (provider) synchronously
  - ▶ Node will block after sending the request until the response is received

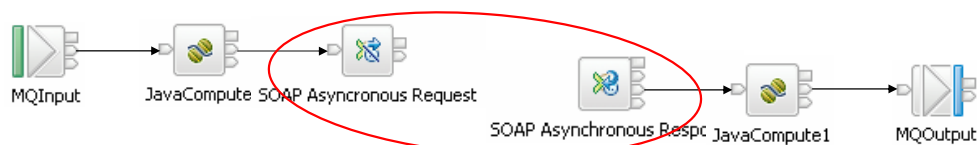


The next new node is the SOAP request node. This is used from within a message flow to invoke a Web service that executes outside the broker environment.

It is a synchronous request, which means that the message flow waits for a response from the Web service, before proceeding with the rest of the flow. Data is passed as SOAP messages, and the data passed back from the SOAP request node can then be accessed and propagated down the remainder of the message flow.

## Asynchronous SOAP request and response

- SOAP AsyncRequest and SOAP AsyncResponse nodes
  - ▶ SOAP AsyncRequest issues request and continues processing
    - Does not wait for the associated Web service response
  - ▶ SOAP AsyncResponse node in a separate thread wait for responses
    - Responses may be in different order than requests
  - ▶ Node correlator identifies logical pairing of async request and response nodes
  - ▶ Reference parameters pass state between request and response SOAP messages



10

Web services SOAP nodes

© 2008 IBM Corporation

The last two SOAP nodes are the asynchronous request and asynchronous response nodes.

In this case, the SOAP asynchronous request node is responsible for sending the Web service request. However, it is not responsible for receiving the response from the Web service. The response is handled by the SOAP asynchronous response node. The asynchronous response node runs on a separate thread from the request node, even if the response component is in the same message flow as the request component.

Using this design, the responses to asynchronous requests can be received in a different order from the original requests. The SOAP asynchronous request node does not perform a blocked wait when it issues the service requests. Therefore, the message flow could issue a large number of asynchronous requests to a service provider, before any responses are received from the provider. The provider can also process these requests in any order, and the responses will be sent back to the message flow, again in any order. This can depend on many variables, such as the complexity of the service request, or a service level agreement delivered by the service provider.

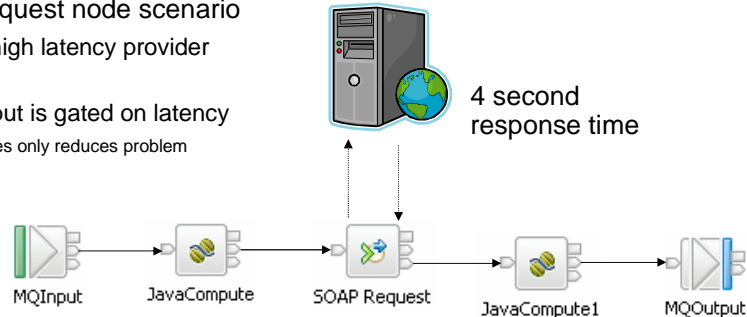
The request and response nodes are paired together by using the node correlator; this will be described in more detail later in this session.

The request and response nodes for a particular application do not need to be in the same message flow. However, they do need to be in the same broker execution group.

## SOAP AsyncRequest and response nodes

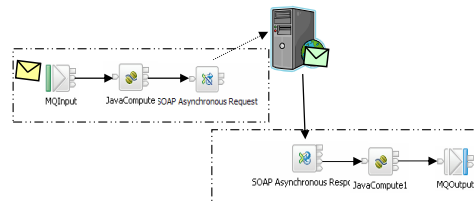
- Synchronous Request node scenario

- ▶ Not optimal for high latency provider scenarios
- ▶ Overall throughput is gated on latency
  - Additional instances only reduces problem



- Asynchronous Request node scenario

- ▶ Thread released once request has been sent
- ▶ Does not block waiting for a response
- ▶ Multiple requests can be handled in parallel



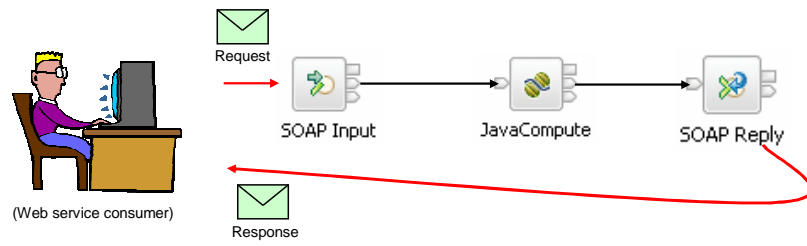
11

This slide illustrates the difference between the synchronous SOAP request node, and the asynchronous request and response nodes.

At the top of the slide, the SOAP request will wait until the response from the provider is received. The message flow will perform a blocked wait until the response is received, or until a timeout occurs. This results in longer response times and a reduced processing capability if the number of thread instances approaches the maximum.

At the bottom of the slide, the same scenario using the asynchronous request and response nodes shows how these can be used in combination to achieve a higher throughput, and possibly faster response time.

## Web service provider flow scenario



12

Web services SOAP nodes

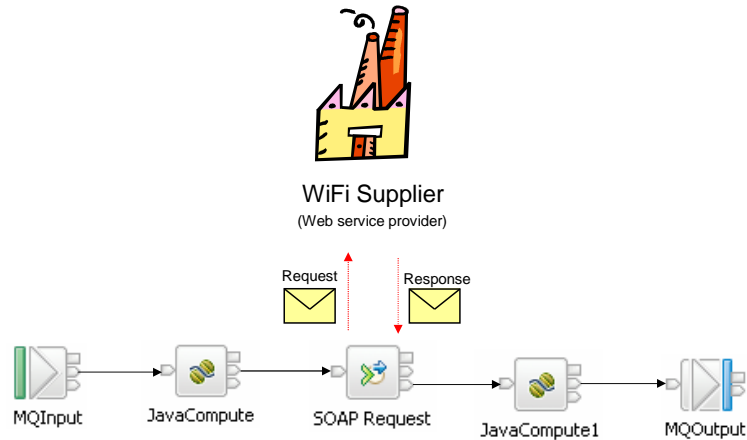
© 2008 IBM Corporation

Using the SOAP and WSDL descriptions discussed earlier, the next few slides now show some examples of how these new nodes might be used in a Message Broker application.

This slide shows a message flow, acting as a Web service provider. The Java compute node is the core of the Web service, and the SOAP input node is the way that the message flow is initiated. The SOAP reply node sends the response of the Web service back to the consumer.

The request from the consumer, shown in the green envelope, is the SOAP message, followed by the outbound SOAP response.

## Web service consumer flow scenario



13

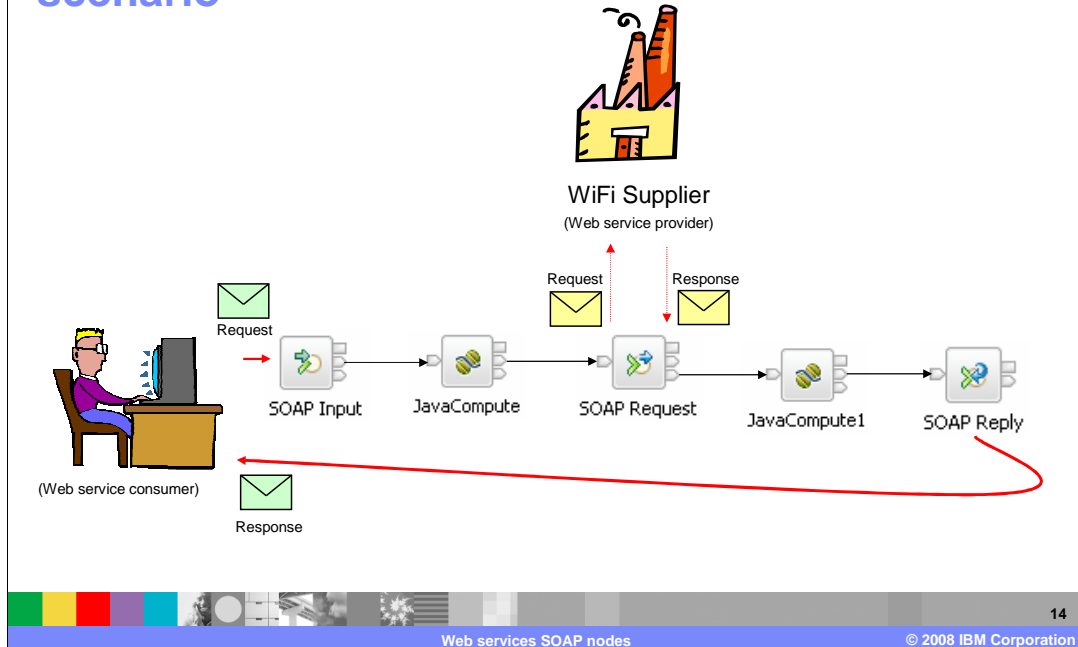
Web services SOAP nodes

© 2008 IBM Corporation

This slide shows a message flow making a Web service request to provider, by using the SOAP request node. Again, requests and responses to and from the provider are carried as SOAP messages.

This scenario uses a synchronous call to a Web service, so the message flow will wait for the response from the Web service provider.

## 'Online retailer' provider and consumer flow scenario



This example combines the previous two, and shows a Web service consumer requesting a service, represented by a message flow. This message flow in turn makes a Web service call to a service provider, and waits for the response from the provider, before completing the response to the original request.

This example only invokes a single service provider. Of course, it could easily be extended to invoke multiple service providers.

## New SOAP nodes - Summary

- Five new 'SOAP' nodes
  - ▶ SOAP input and SOAP reply
  - ▶ SOAP request
  - ▶ SOAP AsyncRequest and SOAP AsyncResponse
- Supports Web service provider and consumer scenarios
  - ▶ Supports request / response
  - ▶ Process SOAP messages
  - ▶ Support both synchronous and asynchronous processing
- WSDL plays a large part in defining and configuring the flow
  - ▶ Drag and drop to create skeleton provider and consumer flows
  - ▶ Configure explicitly on node
- Advanced capabilities configured on nodes
  - ▶ WS-Addressing
  - ▶ WS-Security

This slide summarizes the new SOAP Nodes in Message Broker 6.1.

The two previous slides show the SOAP Input and SOAP Reply Nodes. The SOAP Input Node is used to represent a Message Flow as a Web service. The SOAP Request Node is used to enable a Message Flow to invoke a Web service.

Two further nodes are the Asynchronous Request and Asynchronous Response Nodes. These nodes can be used from within a Message Flow to asynchronously invoke a Web service. This means that the Message Flow does not wait for the Web service response to be provided, but instead can proceed, and the response will be received at some later time. These nodes will be described later in this session.

These nodes enable all types of Provider and Consumer scenarios, covering Request/Response applications, and using either synchronous or asynchronous processing. In all cases, SOAP messages are used to carry the request or response, and these are the only type of messages that are supported by the SOAP nodes.

These applications are developed using WSDL definitions. If the WSDL definitions of a Web service are available, this can be imported into the Message Broker Toolkit, and then dragged onto the message flow pane. The "drag-and-drop" operation will result in a pop-up window, which will allow you to specify whether this is provider scenario, or a consumer scenario. Depending on the response at this time, the toolkit will generate appropriate components of the message flow.

The Web services functions also support advanced Web services standards, such as WS-Addressing and WS-Security. These are specified using properties of the SOAP nodes in a message flow. They are fully described in other sessions.

## Section


# ***SOAP nodes - Properties***

This section discusses the more detailed implementation of the SOAP nodes, and looks at the properties that can be set on the nodes.



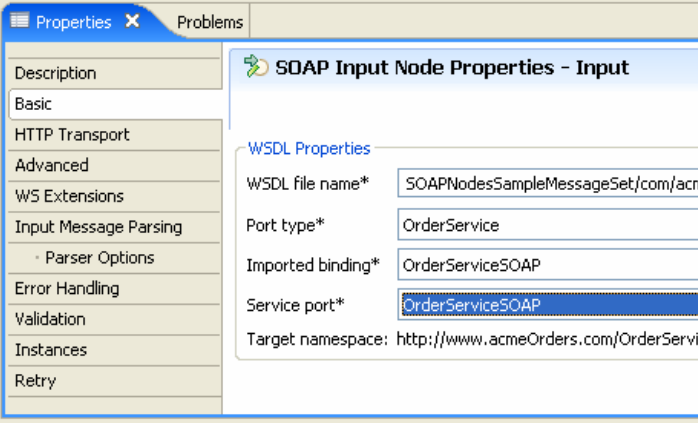
IBM Software Group IBM

## SOAP input node properties



SOAP Input

- The properties of the node are organized under these tabs
  - ▶ **Description: node name, short and long descriptions**
  - ▶ **Basic**
  - ▶ **HTTP Transport**
  - ▶ **Advanced**
  - ▶ **WS-Extensions**
  - ▶ **Input message parsing**
    - Parser Options
  - ▶ **Error handling**
  - ▶ **Validation**
  - ▶ **Instances**
  - ▶ **Retry**
- Has standard input node terminals



The screenshot shows the 'SOAP Input Node Properties - Input' dialog box. The 'Basic' tab is selected in the left-hand pane. The right-hand pane contains the following fields:

- WSDL file name\*: SOAPNodesSampleMessageSet/com/acm
- Port type\*: OrderService
- Imported binding\*: OrderServiceSOAP
- Service port\*: OrderServiceSOAP
- Target namespace: http://www.acmeOrders.com/OrderService

17

Web services SOAP nodes © 2008 IBM Corporation

This slide discusses the properties of the SOAP Input Node. The key properties are those highlighted in red.

On the basic tab, you configure WSDL-related properties, in particular: WSDL file name, port type, binding, and service port. This tab also displays the target namespace, and a list of operations defined by the port type. These are set for you from the associated WSDL.

If the WSDL contains more than one binding or port type, then the correct one can be defined from a drop down list.

Multi-file WSDL is supported.

At runtime, if the SOAP message contains an operation not defined within the WSDL, a SOAP fault will be returned.

On the HTTP tab, you configure HTTP transport-related properties. These specify the URL Selector for this Web service, whether to use HTTPS, and what Maximum Client wait time should be used.

The Advanced tab allows you to specify the 'advanced' properties of "SOAP role", "Set destination list", "Label prefix", and "Must understand" headers.

The SOAP specification requires a SOAP processor to create a SOAP fault if it receives a SOAP envelope that contains a SOAP header that the processor does not understand. The processor must understand the meaning assigned to that header and know how to process it. However, in the case of Message Broker, the SOAP processor is actually the message flow. Hence, there needs to be a way to tell the node what SOAP headers the flow itself will "process" or "handle". This means that the node will create a SOAP fault message if it receives a message that contains a header that it does not understand.

The WS-Extensions tab is used to specify the use of WS-Addressing and WS-Security.

XPath expressions with an associated alias value are added to the WS-Security table. The alias is resolved in a policy set that is created by the administrator. The policy set resolves the alias to either encrypt or sign the part of the message referred to by the XPath expression.

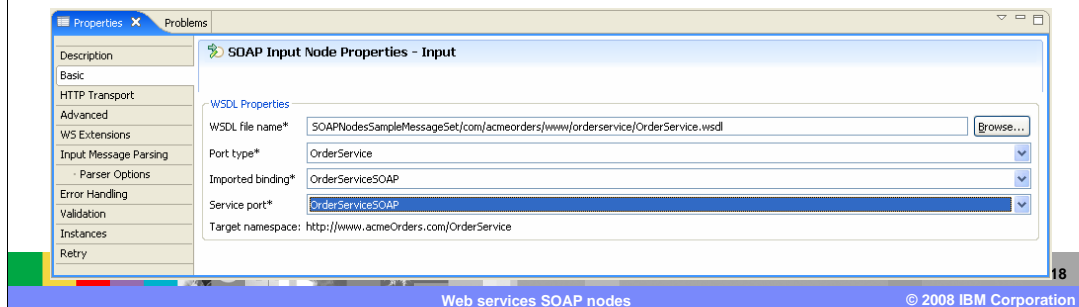
Both WS-Addressing and WS-Security are covered in separate sessions.

The error handling tab is used if a SOAP fault results during inbound SOAP processing. Instead of immediately sending the SOAP fault back to the client, the SOAP fault can be sent to the failure terminal to allow logging and recovery processing. In this situation an exception list is sent to the failure terminal, with the inbound message as a BLOB.

The default value of this property is false, so the default action is to return the SOAP Fault to the Web service client.

## SOAP input node – Basic tab

- The node is configured through a WSDL file
  - Specific node properties are taken from WSDL
  - Reduces configuration errors
  - Drag and drop and explicit configuration supported
- Drag-and-drop support
  - (or select from a node property) the WSDL file that provides the configuration information.
- Allows you to configure the node in accordance with the Web service definition
  - Supported operations
    - Validate the incoming message
  - Defines the format of the incoming request (SOAP, SwA, MTOM), ensuring that it is converted to a standard tree shape (in SOAP domain) format.



This slide shows the details on the basic tab.

You can select the port type and binding that you want to use.

Multi-file WSDL is supported. In this case, the WSDL is split up into several files. Note that that the file that is dropped onto the node must contain the binding or service part of the WSDL.

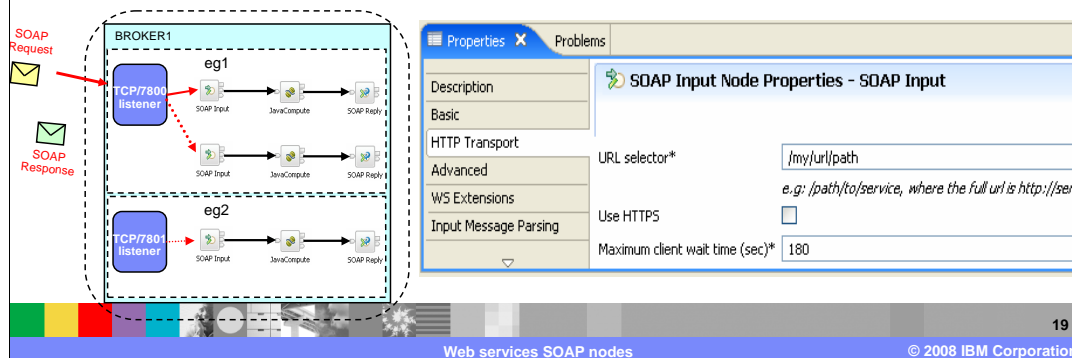
The service port property the required port type.

The port type contains the operations that are supported by this SOAP input node

In Message Broker Version 6.0, the HTTP nodes could be used to service a Web service request directed at the same URL, and hence the same HTTP input node. As an example, assume that the first request was SOAP and the second, related, request was SOAP with an attachment. To handle this scenario in version 6.0, the message flow needs to accept the request as a BLOB and then work out whether to parse it as XML, or as MIME followed by XML. In version 6.1, the SOAP domain will accept both formats and generate the same shape of message tree for each.

## SOAP input node – Http transport tab

- Every execution group containing a SOAP Input node is allocated a TCP/IP port allowing incoming HTTP Requests to be accepted
  - Default port range configured at the broker level
- A SOAP Input node is uniquely addressed using a URL
  - A URL selector defined on the node forms part of the overall URL
  - Combined with the IP address and port number, this allows clients to send messages to a specific SOAP Input node
    - `http://<IP address>:<execution group port>/<URL Selector>`
- The node can be configured to accept HTTP or HTTPS requests



The initial support for the SOAP nodes is with HTTP.

Every execution group containing a SOAP Input node is allocated a TCP/IP port allowing incoming HTTP Requests to be accepted.

When you deploy a message flow which contains a SOAP input node into an execution group, and it is the first SOAP input node to be deployed, then the broker allocates a port number to that execution group. This allows subsequent requests to be properly serviced.

The SOAP input node is uniquely addressable by URL, and the URL selector enables this to be set. The full URL is the IP address of the host machine, the port number of the execution group, and the specific path. The port number is retained by the execution group, even after restart of the execution group.

The port number is created by the broker, and cannot be overridden.

## SOAP input node - Authentication

- Basic authentication is supported by the SOAP Input node and message flow
  - ▶ Verifies a username and password passed within the HTTP header part of the message
- Broker security manager is used to verify user ID and password information.
- Alternative capability is provided with WS-Security



Security profile is now set on the bar file editor


You need to configure security within the broker to exploit this function.

If the security profile has not been set, then any username and password specified will be accepted.

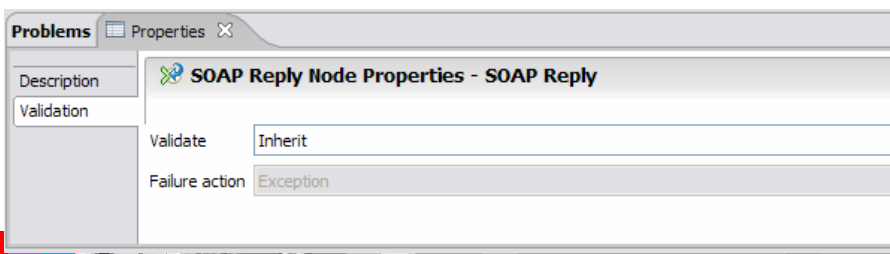
If security profile is set to 'No security', username and password are presumed to be correct; no checking is performed.

IBM Software Group IBM

## SOAP reply node



- It takes responsibility for sending the response message back to the originating client
  - ▶ Designed to be the other-half of a SOAP input node
    - Cannot be used in isolation
    - There must be a SOAP input node somewhere else in the same execution group
  - ▶ Takes certain configuration from associated SOAP input node
    - Entirely dependent upon the incoming message
    - Reference to the originating SOAP input node is passed in the message



The screenshot shows the 'SOAP Reply Node Properties - SOAP Reply' dialog box. It has a 'Description' tab selected. Under the 'Validation' section, there are two fields: 'Validate' with a dropdown menu set to 'Inherit', and 'Failure action' with a dropdown menu set to 'Exception'.

21

Web services SOAP nodes © 2008 IBM Corporation

The SOAP reply node is used in conjunction with the SOAP input node. It responds to a client request which has sent its request using a SOAP input node. It cannot be used in isolation. A message flow that contains a SOAP reply node cannot be deployed to an execution group, if it does not have an associated SOAP input node.

It is possible however, to have one SOAP reply node which handles the replies from more than one SOAP input node within a message flow. To enable this linking of SOAP input nodes to SOAP reply nodes therefore, the broker maintains this information internally.

If a message is received that is not in the SOAP domain, it is assumed to be the SOAP body part of the message. The SOAP reply node will look within the local environment to see if the envelope exists. If it does, then the body will be joined with the body part of the message. If it does not exist, then defaults will be used.


The result of this is that the SOAP reply node obtains its properties from the associated SOAP input node. There are no explicit properties that can be set on the SOAP reply node.

When constructing a message flow, you need to ensure that you preserve the reply identifier, which is passed within the local environment. If this is not available within a message when propagated to a SOAP reply node, then an exception is created.

IBM Software Group IBM

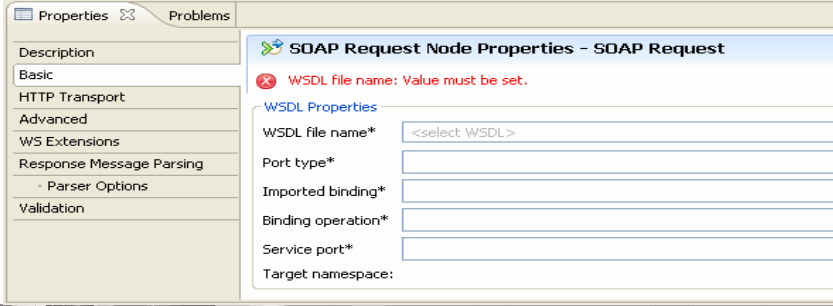
## SOAP request node properties

- The properties of the node are organized using these tabs
  - Description: node name, short and long descriptions
  - **Basic**
  - **HTTP transport**
  - Advanced
  - Response message
    - Parser options
  - Validation



SOAP Request

- Contains 'fault' terminal for SOAP fault



22

Web services SOAP nodes © 2008 IBM Corporation

The presentation now discusses the SOAP request node. The first thing to note on this node is that there is a SOAP fault terminal. This is used in instances where the provider of the SOAP service is unable to handle the SOAP request. In these cases, a SOAP fault will be returned by the SOAP provider, and this will be propagated to the SOAP fault terminal. This should then be connected to an appropriate part of your message flow.

As with the SOAP input node, the SOAP request node is configured using WSDL. WSDL from the project can be dragged and dropped onto the SOAP request node, which automatically configures the key properties of the node. This enables the outgoing message to be generated in the right format, for example a SOAP message with attachments. This action also configures the Web service URL, which identifies the specific protocol address of the Web service provider. However, you can change this subsequently, by editing the generated properties in the normal way.

The remaining properties follow the same approach as for the SOAP input node.

Note that security configuration is performed on the deployment descriptor of the associated bar file, not on the message flow definition.

## SOAP request – Basic tab

- The node is configured through a WSDL file
  - ▶ Specific node properties are taken from WSDL
  - ▶ Reduces configuration errors
- Drag and drop support
  - ▶ (or select from a node property) the WSDL file that provides the configuration information.
- Allows you to configure the node in accordance with the Web service definition
  - ▶ Specifies the “Operation” that must be used
  - ▶ Allows outgoing messages to be generated in the correct format for that operation

The screenshot shows the 'SOAP Request Node Properties - SOAP Request' dialog box. The 'Basic' tab is selected. A red error message at the top states 'WSDL file name: Value must be set.' Below this, the 'WSDL Properties' section contains several fields: 'WSDL file name\*' (a dropdown menu currently showing '<select WSDL>'), 'Port type\*', 'Imported binding\*', 'Binding operation\*', 'Service port\*', and 'Target namespace:'.

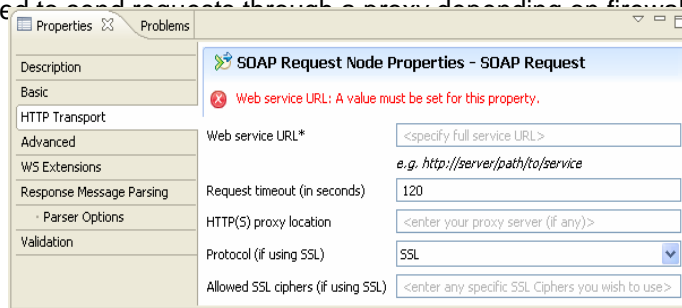
23

The message set belonging to the imported WSDL file will configure the message set property. The imported WSDL file is dropped onto the SOAP request node, and all associated Web services properties are set based on the imported WSDL.

This will reduce the possibility of configuration errors. If you want to change any of the generated properties, this can be done by changing properties in the normal way.

## SOAP Request – HTTP Transport Tab

- The URL of the Web service provider to call is taken from the WSDL file used to configure the node
- A LocalEnvironment override can be specified to override this setting from within the message.
  - ▶ LocalEnvironment.Destination.SOAP.Request.Transport.WebServiceURL
  - ▶ Useful in several environments, including WebSphere Services Registry and Repository scenarios
- Can use an HTTP or HTTPS connection
- Can be configured to send requests through a proxy depending on firewall configuration



24

Web services SOAP nodes

© 2008 IBM Corporation

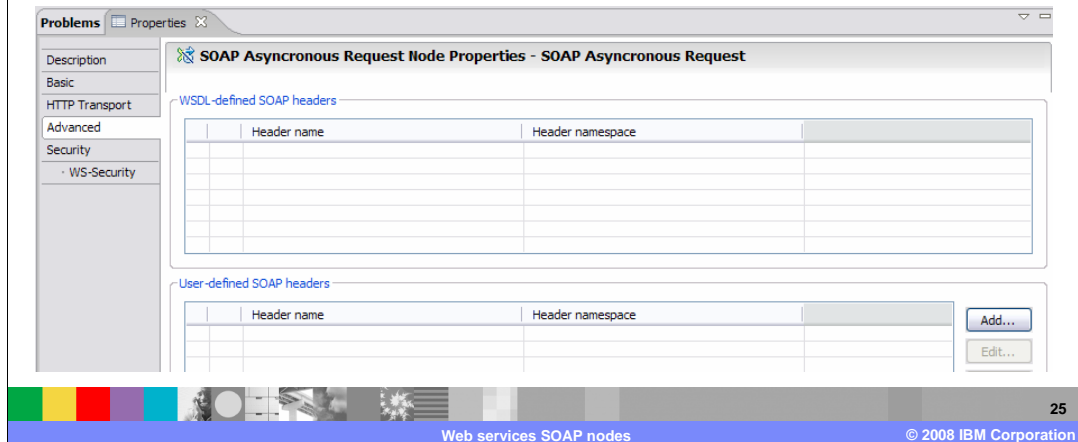
As in the Basic Tab, the HTTP properties are set from the imported WSDL.

The soap address property of the “*selected service*” is used to populate the relevant information on the Transport tab that the “*selected binding*” enabled. For HTTP, the *Url-Selector* property is populated from the address information in the WSDL.



## SOAP request node – Must understand headers

- The “Must understand” headers defined within the **response** message must be defined using the advanced tab
  - ▶ WSDL defined
  - ▶ User defined



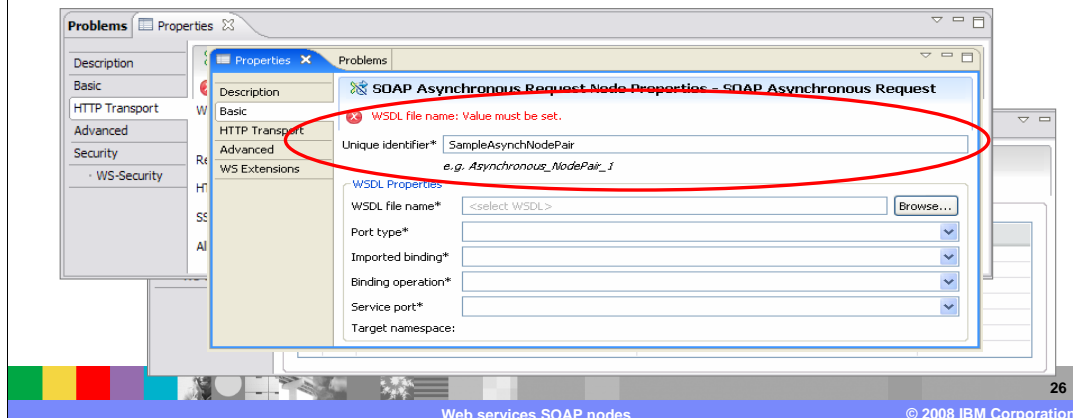
A SOAP message can contain certain headers known as “must understand” headers. These should be defined on the SOAP request node properties, using the Advanced tab. Headers defined as part of the WSDL will be generated automatically. User-defined headers should be added, by clicking on the Add button.

## SOAPAsyncRequest node properties

- Contains all configuration options as the SOAPRequest node
  - ▶ WSDL configuration from drag-and-drop
  - ▶ Contains a correlation ID to connect AsyncRequest to AsyncResponse node
  - ▶ Has 'out' and 'failure' terminals



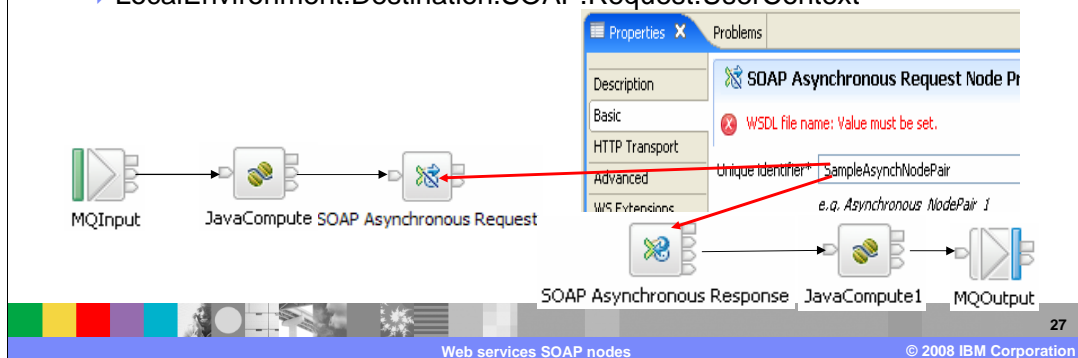
SOAP Asynchronous Request



This slide shows the properties of the asynchronous request node. Most of the properties are the same as the request node. However, in the case of the asynchronous node, you must specify the unique correlation ID which enables the reply to be matched with the request. These ID must be specified on the corresponding reply node properties, as shown on the next slide.

## SOAP async request / response pair

- Nodes are 'paired' through a Correlation ID (shared key)
- The SOAP AsyncResponse node is effectively an Input node, because it starts a new thread
  - ▶ SOAP AsyncResponse node starts a new transaction
  - ▶ If an exception occurs in second flow (containing SOAP AsyncResponse node), message is not rolled back to first flow
  - ▶ SOAP Fault messages will be directed down the Fault terminal
- You can store a single piece of context data in the local environment
  - ▶ LocalEnvironment.Destination.SOAP.Request.UserContext




The SOAP asynchronous request and response nodes are connected through the use of the correlation ID property. This property must be set to an identical string on both nodes. This will typically be part of the associated URL that is being invoked.

Although a SOAP asynchronous response node may appear in the same message flow as the corresponding request node, it will in fact operate more like a SOAP input node. This means that it will start a new processing thread when the response node is activated. In terms of transaction processing, the SOAP asynchronous response node will create a new, separate transaction from the request node. If the SOAP response node processing encounters any processing errors, then these errors will be propagated down the fault terminal of the SOAP response node. However, any changes made by the message flow containing the SOAP request node will not be rolled back.

IBM Software Group IBM

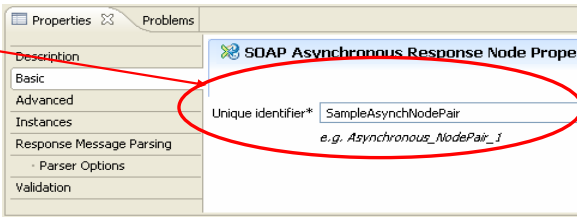
## SOAP AsyncResponse node properties



failure  
out  
fault  
catch

SOAP Asynchronous Response

- The properties of the node are organised under these tabs:
  - ▶ Description: node name, short and long descriptions
  - ▶ **Basic**
  - ▶ Advanced
  - ▶ Instances
  - ▶ Response Message Parsing
    - Parser Options
  - ▶ Validation
- Inherits most of configuration from SOAP AsyncRequest Node
- Contains 'out', 'fault', 'failure' and 'catch' terminals



28

Web services SOAP nodes © 2008 IBM Corporation

When setting the properties of the SOAP asynchronous response node, the majority of the properties are inherited from the associated SOAP request node.

Therefore, it is not necessary to drag the WSDL onto this node.

The only property that must be set explicitly is the correlation ID property.

## SOAP AsyncRequest and SOAP AsyncResponse nodes

- SOAP AsyncRequestNode is in a separate flow to the SOAP AsyncResponseNode
  - ▶ Provider sends response to a different destination to the caller
- WS-Addressing is used to express the return address
- WSA:ReplyTo and WSA:FaultTo headers are set to point to the corresponding response node in the form:
  - ▶ *http://hostname:7800/path/correspondingAsyncResponseNodeID*
- WS-Addressing is supported under both the Web services provider and consumer scenarios
- WS-Addressing is covered in detail in a separate session

29

Web services SOAP nodes

© 2008 IBM Corporation

When using the asynchronous nodes, a way is needed to handle the scenario where the Web service requester wants the response to be sent to a different destination than the requester. The requester therefore needs a way to express the return address.

This is achieved by using the Web services standard, namely WS-Addressing. WS-Addressing can be used by all SOAP nodes, but is required when using the asynchronous nodes. This is done by using the ReplyTo and FaultTo headers. The subject of WS-Addressing is covered in a separate session.

## Summary

- Five new SOAP nodes
  - ▶ SOAP input
  - ▶ SOAP reply
  - ▶ SOAP request
  - ▶ SOAP async request
  - ▶ SOAP async response
- Web service provider and consumer scenarios are supported

This session has discussed the five new SOAP nodes.

The SOAP input and SOAP reply nodes are used in conjunction, and are used to represent a message flow as a service provider. The SOAP input node receives a SOAP request, and the SOAP reply node responds to that request.

The SOAP request node invokes a Web service synchronously.

The SOAP asynchronous request and response nodes invoke a Web service asynchronously. WS-Addressing is used to identify the location of the SOAP response, if this is not the same as the SOAP request.

The message processing and WS-Addressing subjects are covered in separate sessions.

## Feedback

### Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

[mailto:iea@us.ibm.com?subject=Feedback\\_about\\_WMB61\\_IEA\\_WebServices\\_SOAP\\_Nodes.ppt](mailto:iea@us.ibm.com?subject=Feedback_about_WMB61_IEA_WebServices_SOAP_Nodes.ppt)

This module is also available in PDF format at: [../WMB61\\_IEA\\_WebServices\\_SOAP\\_Nodes.pdf](..\\WMB61_IEA_WebServices_SOAP_Nodes.pdf)



You can help improve the quality of IBM Education Assistant content by providing feedback.

## Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM                      WebSphere

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.