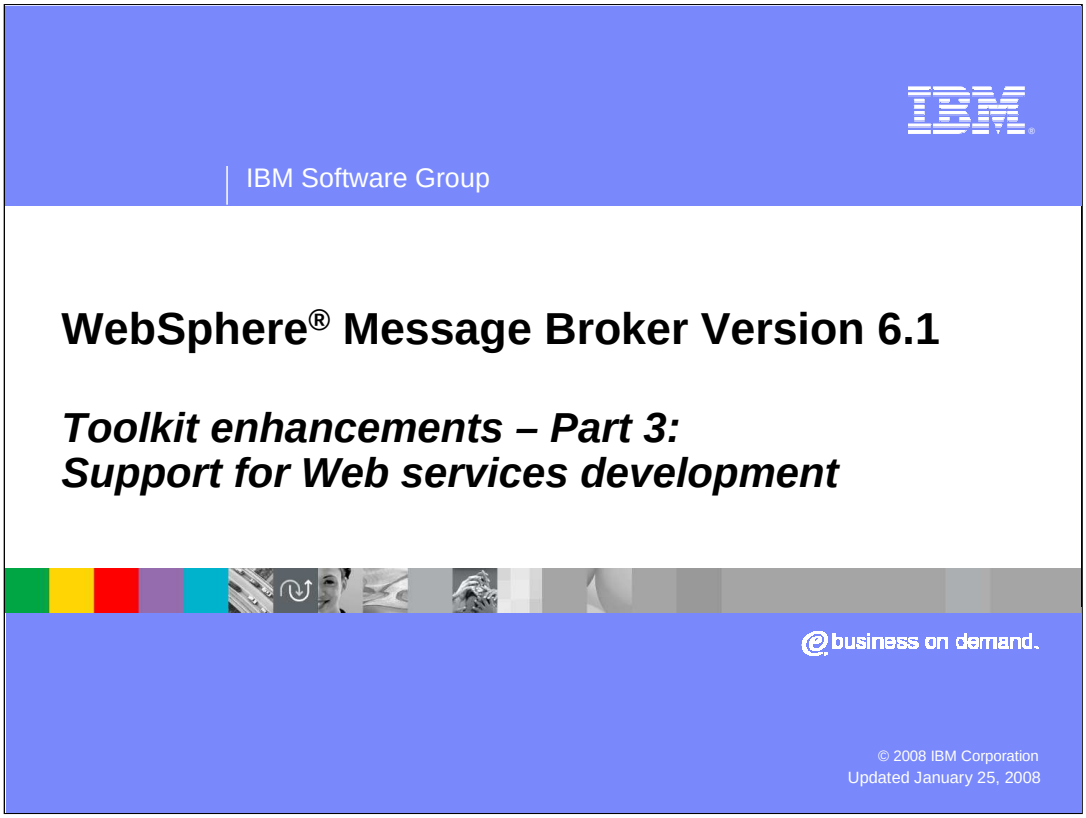
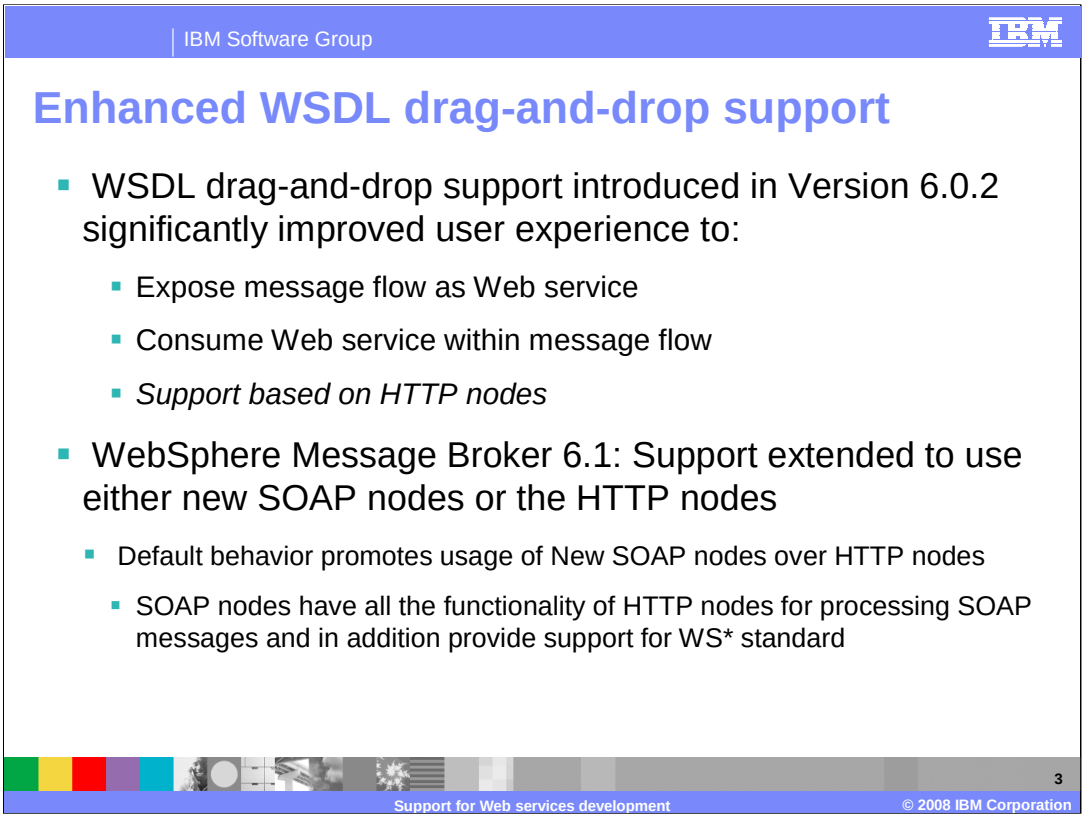


This presentation covers some of the new functions in the toolkit in WebSphere Message Broker Version 6.1.

The primary topics of this presentation are the enhancements made to support application development with Web services.

The presentation will look at the WSDL importer, and how message sets are created and organized. It will look at how WSDL files can be used to automatically populate nodes in a message flow, and how this function is optimized for the new SOAP nodes in version 6.1.

And it will demonstrate this with two scenarios. The first discusses how to expose a message flow as a Web service, and the second is how to invoke a Web service from within a message flow.



As a reminder, Message Broker version 6.0.2, released in December 2006, introduced new function to automate the implementation of Web services development. This function provided the ability to use WSDL to create a new message flow as a Web service, or to invoke a Web service from a message flow. The WSDL could be dropped onto the message flow palette, and depending on which of the two Web service models was selected, the appropriate nodes were populated.

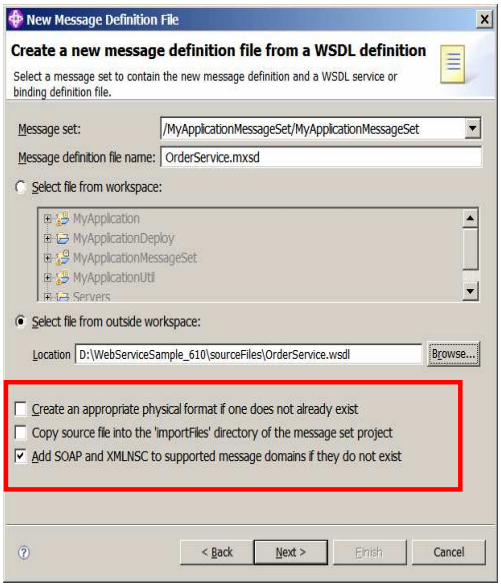
This support was available only for the HTTP nodes.

Version 6.1 extends this support to the new SOAP nodes, but still supports the HTTP nodes. The support for the SOAP nodes is only for the HTTP transport.

WSDL importer and quick start wizard

New options:

- Add SOAP and XMLNSC domains if they did not exist. Checked by default to:
  - **Promote usage of new SOAP nodes** while creating flows through WSDL drag-and-drop
- Save backup copy of original WSDL in ***importFiles*** folder. Not checked by default:
  - Because in MB6.1 WSDL is key artifact of message set, and the modified copy of imported WSDL is stored inside the message set folder
  - Also reduce confusion as to which WSDL file should be used because there are two copies of WSDL – the original and the imported version - in the message set project.
- Create XML wire format if one does not already exist provided MRM is one of the supported domain on the message set.



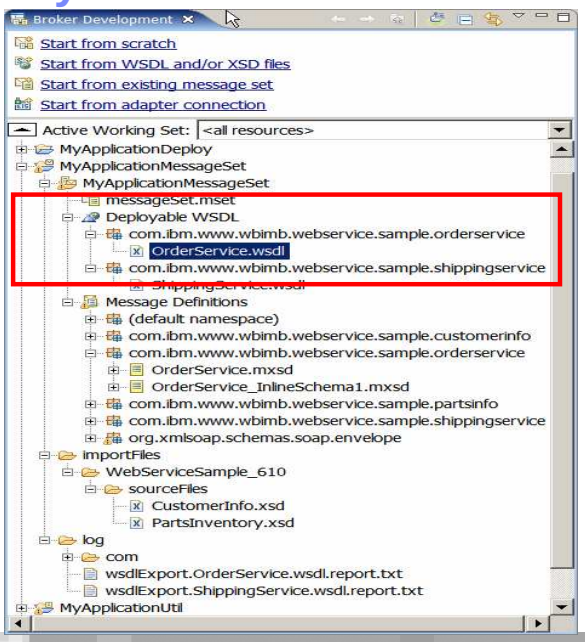
This slide shows the window that appears when WSDL is imported into the toolkit. The default action is to add the SOAP and XMLNSC domains as supported domains in the resulting message set. This is shown highlighted on the screen capture. The intent is to promote the use of the SOAP nodes, although you can change this if required.

The second key option is to specify whether the original WSDL should be copied into the project workspace, in the “importFiles” folder. In this case, the default is not to do this. This is because the WSDL itself forms part of the message set, and the presence of two copies of the WSDL in different parts of the same message set could result in confusion. However, you can override this action if necessary.

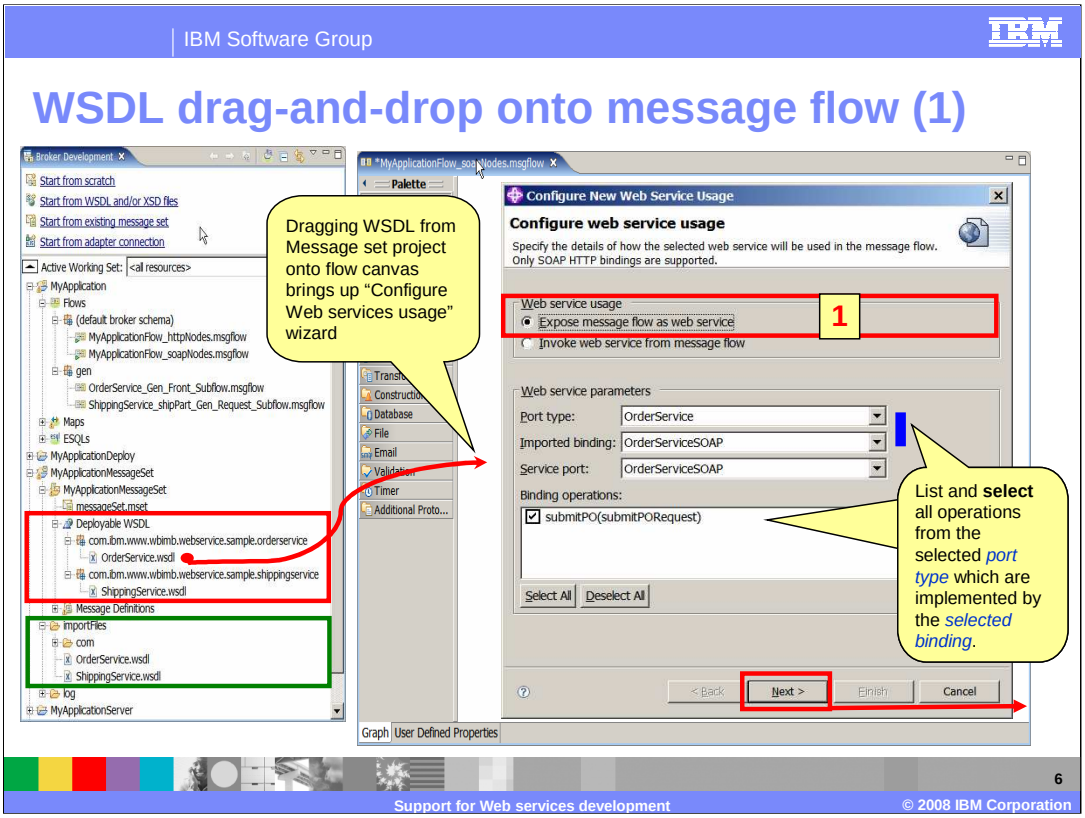
Finally, if the domain is “MRM”, an XML wire format is created by default.

Message Set - WSDL key artifact

- WSDL files imported into message set are
  - Stored in subfolder derived from nsURI of WSDL
  - Shown under new logical category "Deployable WSDL"
  - Schema type definitions referenced externally or defined in-lined are changed to reference mxsd files in the message set
  - WSDL files are annotated to
    - Keep track of mxsd files created by WSDL importer.
    - Keep WSDL files in synch with MXSD files



As mentioned on the previous slide, WSDL is now a key artifact of the message set, and is stored in the new category called "Deployable WSDL". The name of the sub-folder is derived from the "nsURI" of the WSDL.



Now that the message set has been created, and the appropriate WSDL imported into the message set, the next task is to create a message flow.

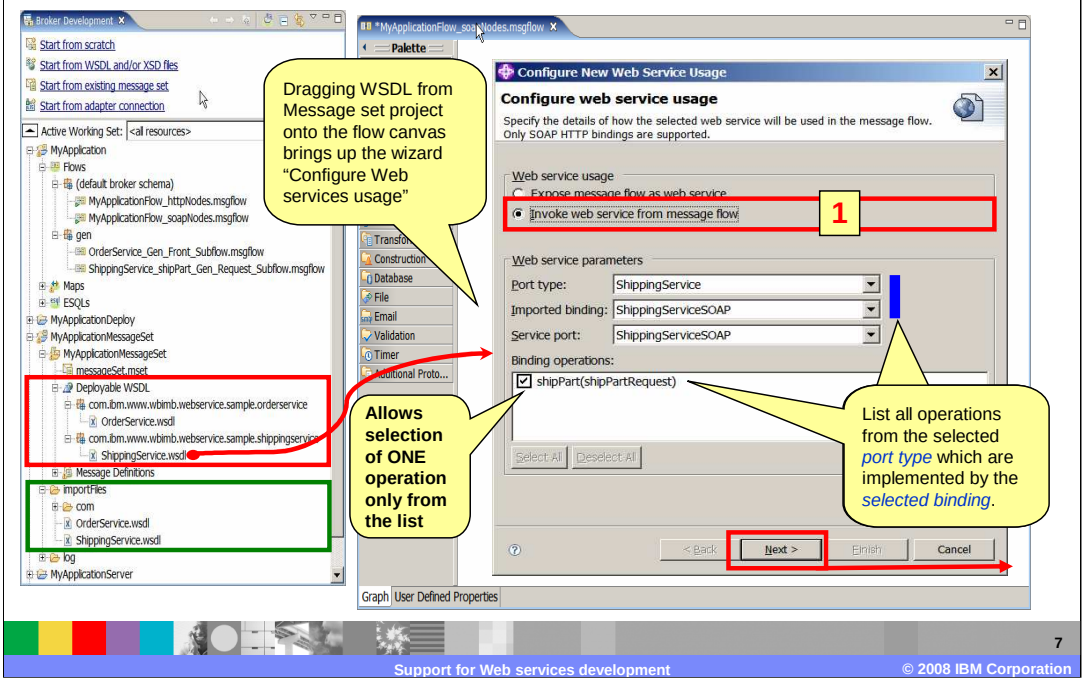
On this screen capture, the WSDL that is used in the message flow is shown under the "Deployable WSDL" category. Note also that the WSDL has been stored in the "importFiles" category. However, these files are not used in the creation of the message flow.

So, when the message flow has been created, the first action is to drop the WSDL onto the flow canvas. At this point, a window similar to the one shown on this slide opens. This window asks you to specify whether the flow should be exposed as a Web service, or whether the WSDL should be used to invoke a remote Web service. This is shown highlighted as number 1 on this slide.

Depending on which option you select, the wizard automatically selects the binding operations, and the service port that should be used. If multiple bindings and ports are available, these are all listed, and you must make an appropriate selection.

The example shown on this slide shows the "OrderService" port type, and the "submitPO" biding operation.

WSDL drag-and-drop onto message flow (2)



This slide shows a similar operation, except in this case, the selection is to invoke a Web service from the message flow. This is shown again as number 1 on this slide.

The operation is now the "ship-Part" operation, since the WSDL that has been dropped onto the canvas is the "ShippingService" WSDL file.

Select nodes to use for generated flow

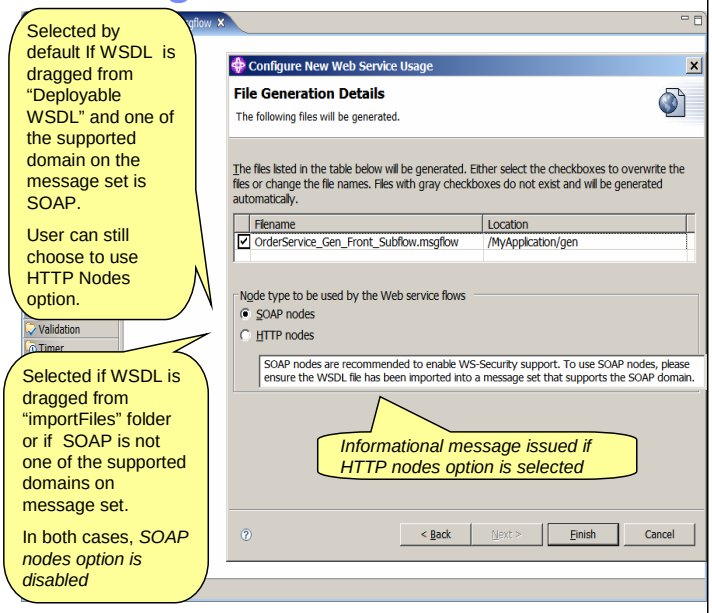
Wizard behavior geared towards **using new SOAP nodes** for the generated flow/sub flow

Default node selection determined from :

- From where in the message set project, WSDL file is dragged
- And a set of domains supported on the message set

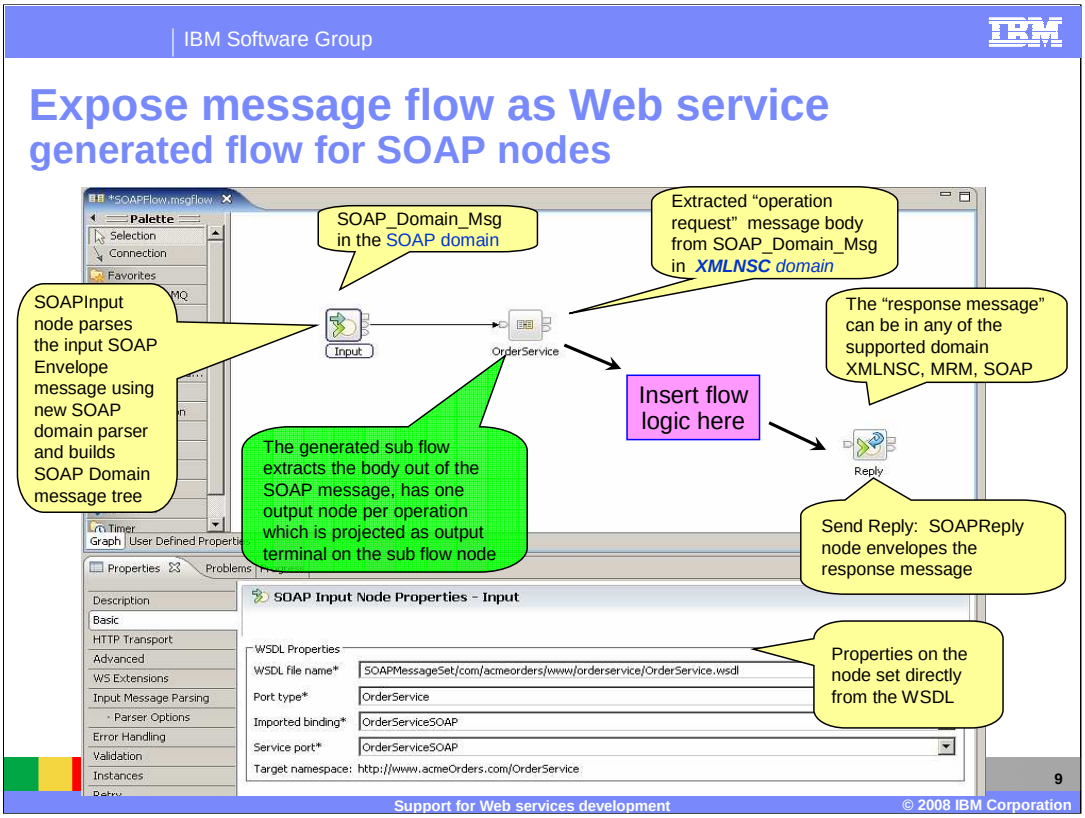
Backward compatibility

- Message sets created with 6.0.2 may have WSDL files in ***importFiles*** folder.
  - WSDL dragged from importFiles folder have HTTP nodes option only



The next stage is to select which type of nodes will be used to support this Web service. The default nodes are the SOAP nodes, and this can be changed to use the HTTP nodes. The SOAP nodes require the specification of the SOAP domain on the associated message set. This is done automatically when the WSDL is imported into the message set project.

If you are using this function with message sets that have been created using Message Broker Toolkit version 6.0.2, then the WSDL files will be located in the "import Files" folder. If you use the WSDL from this location, then the SOAP nodes will be disabled, and you will only be able to use the HTTP nodes. You will not be able to use the SOAP nodes in your message flow.



In the next stage, the wizard creates a skeleton message flow, with several nodes. The example on this slide shows a message flow being exposed as a Web service.

The first node is a SOAP input node. This is named “Input” by default. The properties of the input node are populated automatically, using the values derived from the imported WSDL, and the selections that were made in earlier stages of the wizard. Hence, the port type, binding and service port are all generated by the wizard. The SOAP input node creates a SOAP domain message, based on SOAP messages that arrives at this input node.

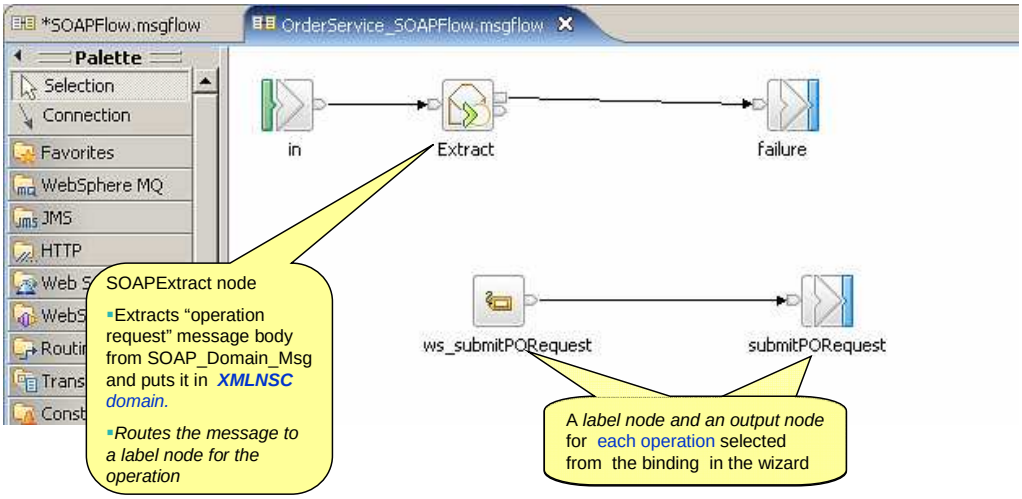
The second node, called “OrderService” invokes a subflow. This subflow exposes a terminal for each of the operations that were selected at the previous stage of the wizard. This subflow extracts the message body, and removes the SOAP envelope. The output from this subflow is the payload of the message, and is in the XML-NSC domain.

This subflow is covered in more detail on the next slide.

The final node is a SOAP Reply node, which sends a response message, corresponding to the SOAP Input node.

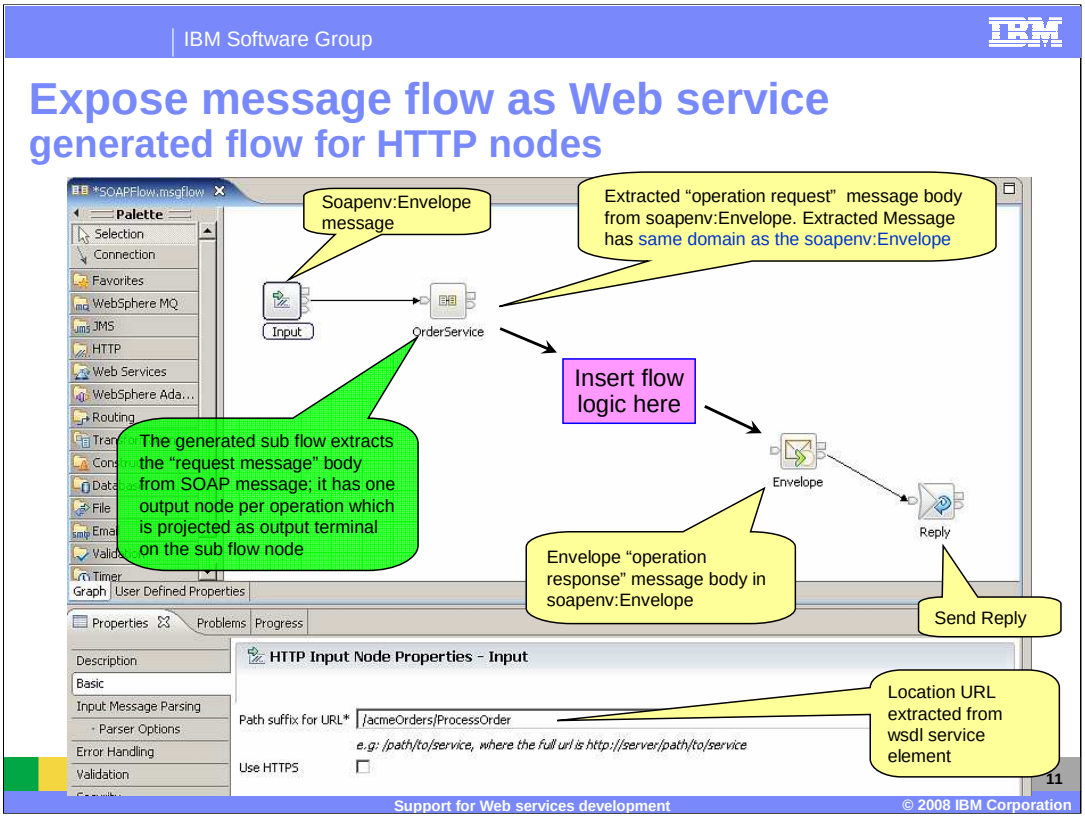
User flow logic is placed in between the generated subflow and the final Reply node.

Expose message flow as Web service
generated subflow for SOAP nodes



This slide shows the generated subflow that is used the handle the incoming SOAP message. The subflow uses a "Route to label" node to handle each operation that was specified in the wizard. The example on this slide has just specified one operation, and therefore only has one Label node. The Label node then passes the message to a corresponding Output node, which in turn corresponds to the appropriate output terminal on the subflow node.

The node named "Extract" is a "SOAP Extract" node. The SOAP Extract and SOAP Envelope nodes are now built into the toolkit in version 6.1. The extract node therefore removes the SOAP envelope, and places the resulting payload of the incoming message into the XML-NSC domain.

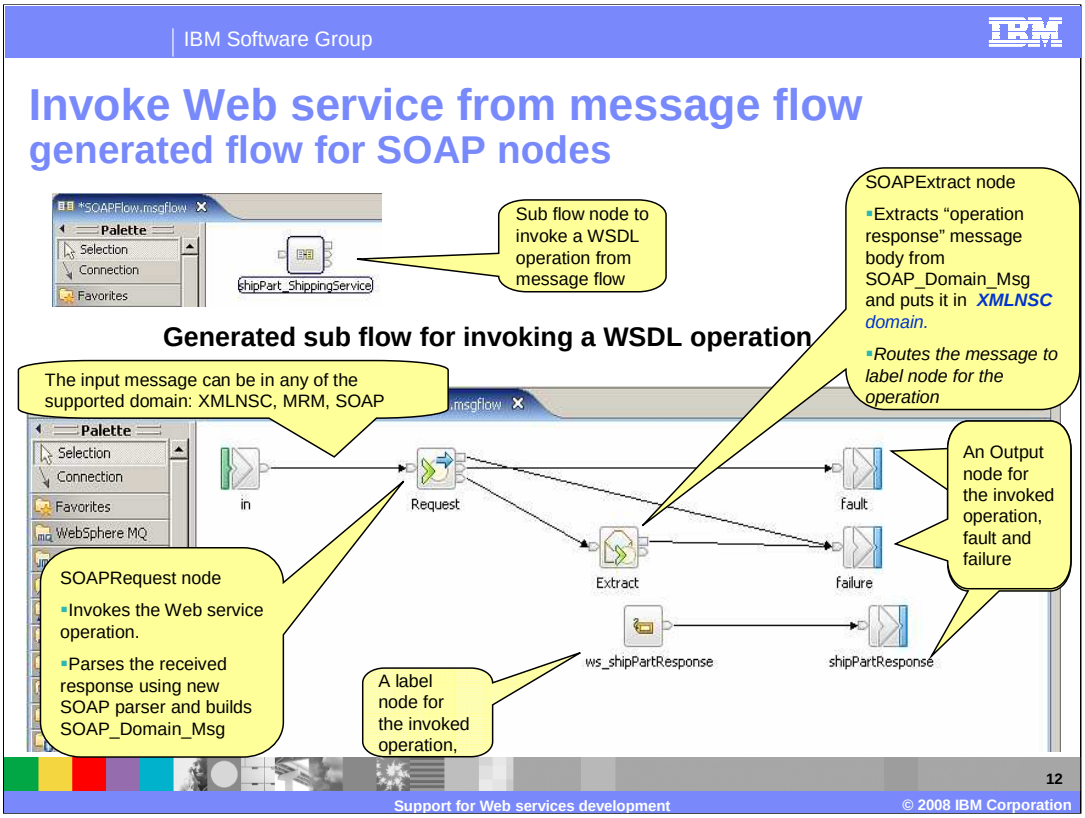


This example shows a similar process, except the SOAP nodes are replaced by HTTP nodes. In this case, the input node is an HTTP Input node, with a corresponding HTTP Reply node.

Note that in this case, a SOAP Envelope node is also required. This is because the HTTP Reply node does not automatically generate a SOAP envelope, whereas the SOAP Reply node does include this.

The generated sub-flow, OrderService, performs the same function as with the SOAP Nodes case. However, in the HTTP case, the input and output from this sub-flow will be in the same domain. This will be determined by the domain specified on the HTTP input node.

This approach is unchanged from the approach used in Message Broker version 6.0.2.



The presentation now discusses the second scenario, where the message flow invokes a Web service.

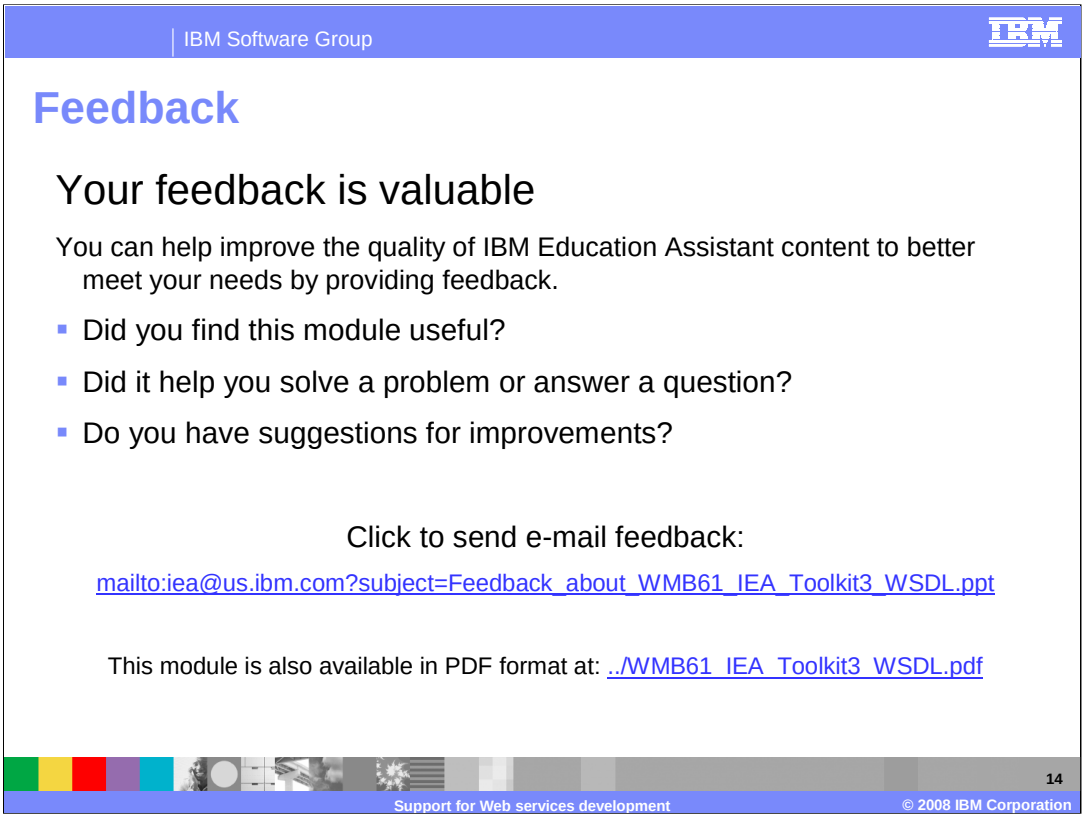
On this slide, the wizard specified the option to invoke an external Web service. This has resulted in the generation of a sub-flow, named “shipPart Shipping Service”. Expanding this sub-flow shows a SOAP request node. The default name of this node is “Request”. On completion, this node passes a message in the SOAP domain to a SOAP Extract node, which removes the SOAP envelope and passes the payload of the SOAP message to the rest of the message flow. The output from the Extract node is in the “XML NSC” domain.

In this scenario, there will only be one label node, along with output nodes to handle SOAP Faults and failures.

Summary

- Toolkit enhancement to support Web services development
- WSDL importer and message set structure
- WSDL drag-and-drop support
- Scenarios
  - Expose message flow as Web service
  - Consume Web service within message flow

In summary, this presentation has shown how to use the drag-and-drop features of the Toolkit to create the artifacts needed for Web services application in version 6.1. It has discussed the structure of the message set definitions, and illustrated this with two scenarios.



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 NONINFRINGEMENT. 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.