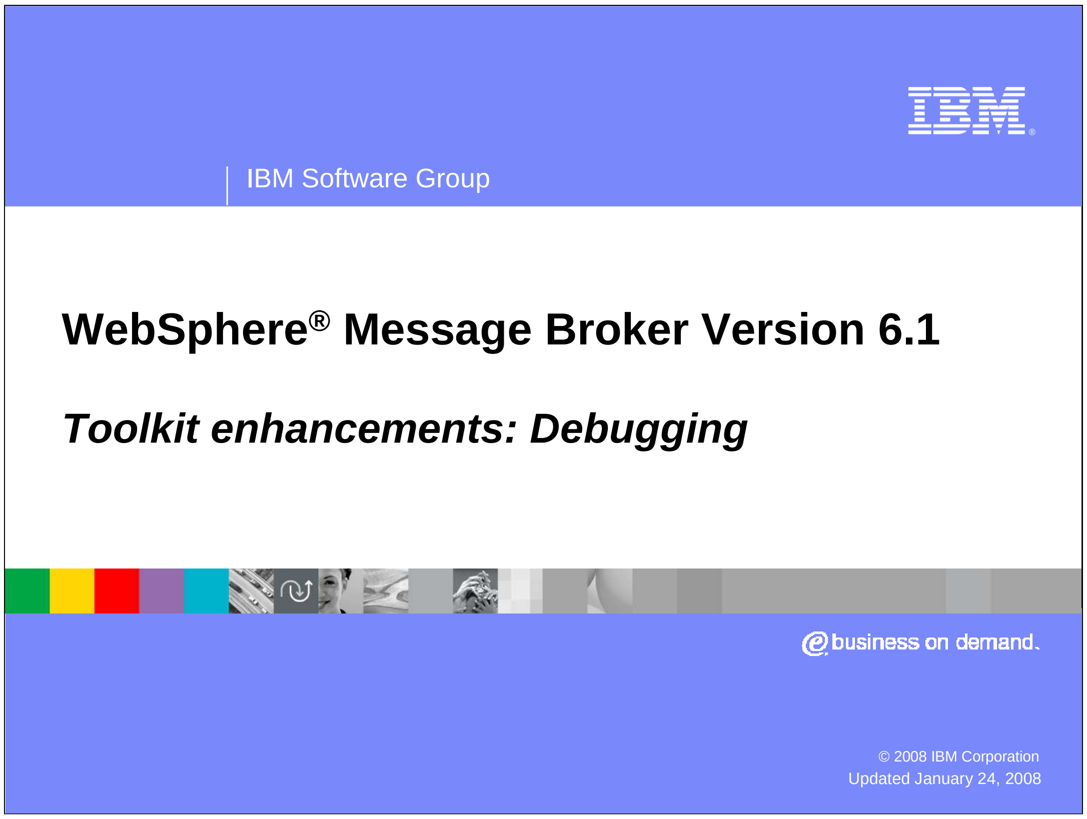
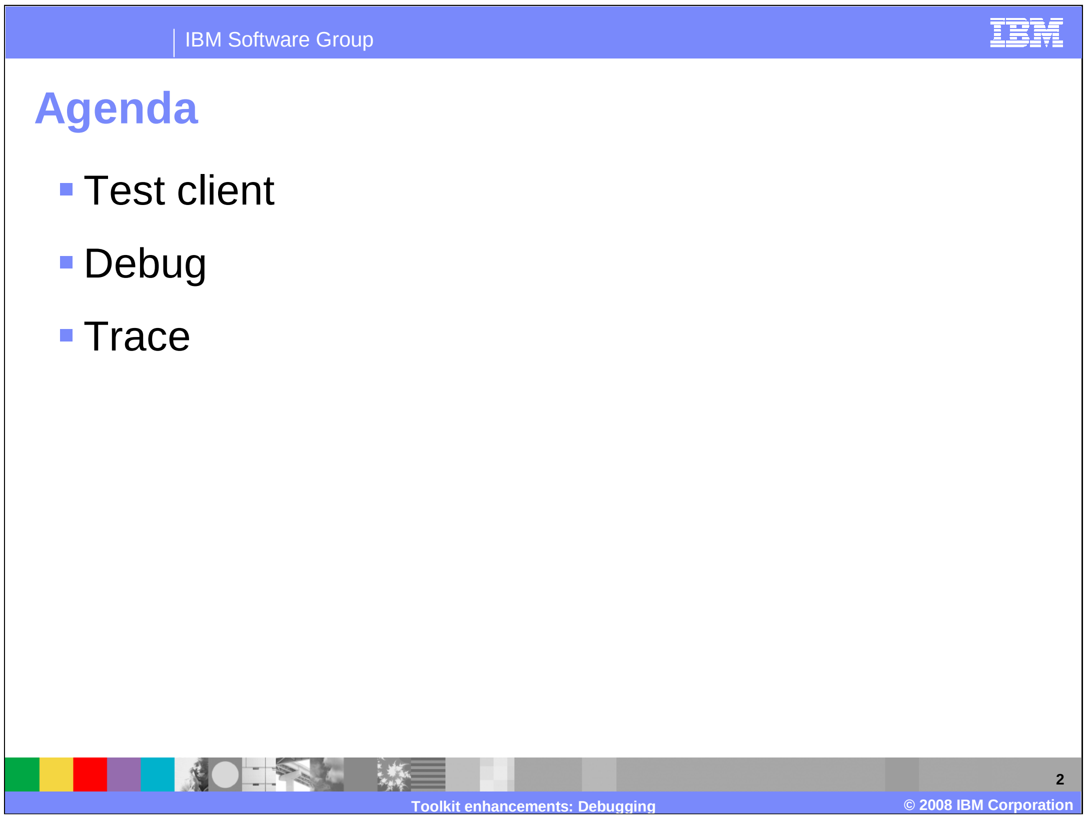


WebSphere® Message Broker Version 6.1

Toolkit enhancements: Debugging

Updated January 24, 2008

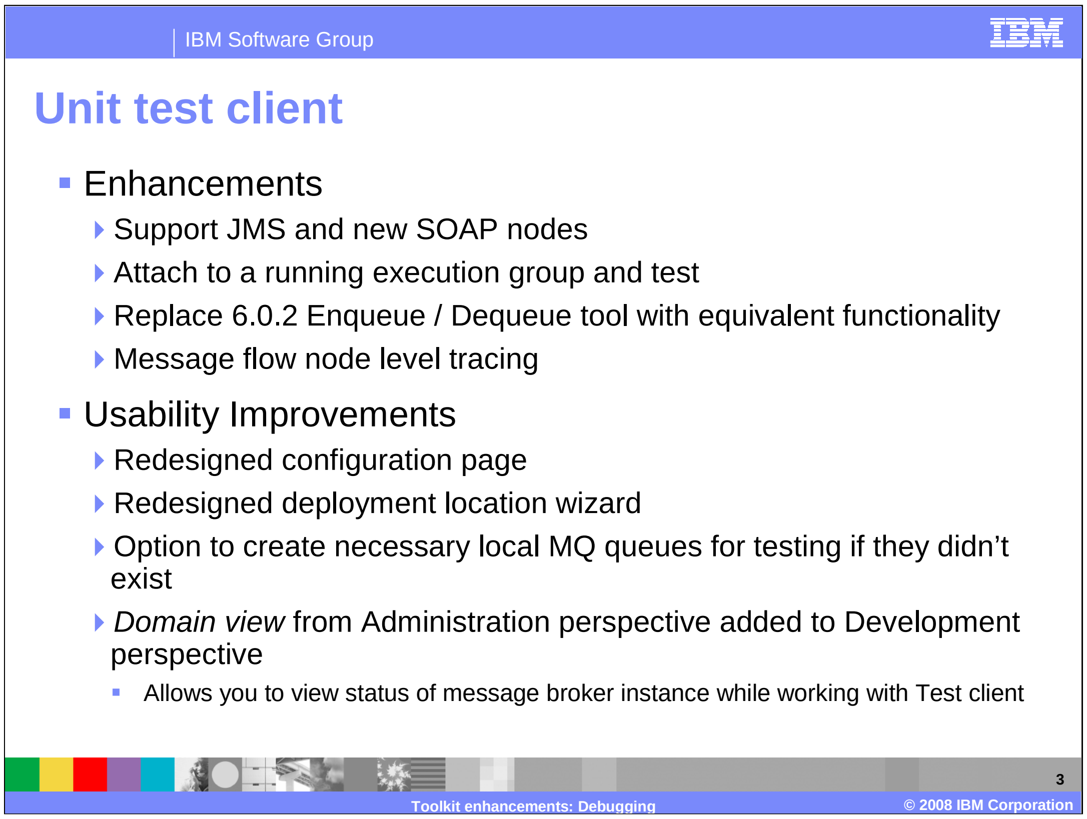
This presentation covers the test tools that are provided within the Message Broker Toolkit.



Agenda

- Test client
- Debug
- Trace

This presentation is broken down into three areas, the Test Client, the facilities for debugging, and new tools for tracing message flows.



Unit test client

- Enhancements
  - Support JMS and new SOAP nodes
  - Attach to a running execution group and test
  - Replace 6.0.2 Enqueue / Dequeue tool with equivalent functionality
  - Message flow node level tracing
- Usability Improvements
  - Redesigned configuration page
  - Redesigned deployment location wizard
  - Option to create necessary local MQ queues for testing if they didn’t exist
  - *Domain view* from Administration perspective added to Development perspective
    - Allows you to view status of message broker instance while working with Test client

The unit test client was introduced in version 6.0.2 of the Message Broker Toolkit. In that release, the unit test client was supported for the HTTP and MQ input nodes. Version 6.1 has extended this support to JMS input nodes and the new SOAP nodes. It has also been enhanced to remove the need to deploy a bar file every time a message flow was tested. In version 6.1, you can use the test client to test a message flow that has been deployed to an execution group in the normal way.

The Enqueue/dequeue support available in version 6.0.2 has been integrated into the test client in version 6.1, and a new function called “node level tracing” has been introduced. These are covered in more detail on later slides.

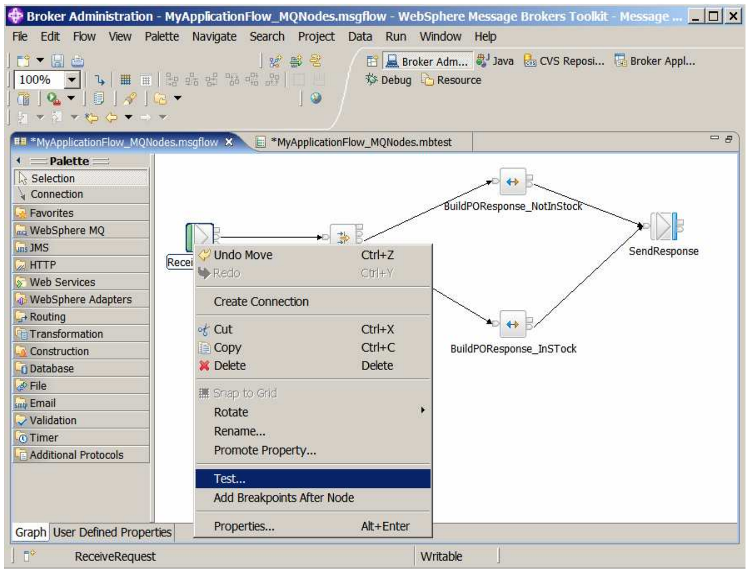
Version 6.1 has also provided several usability improvements.

Invoke test client

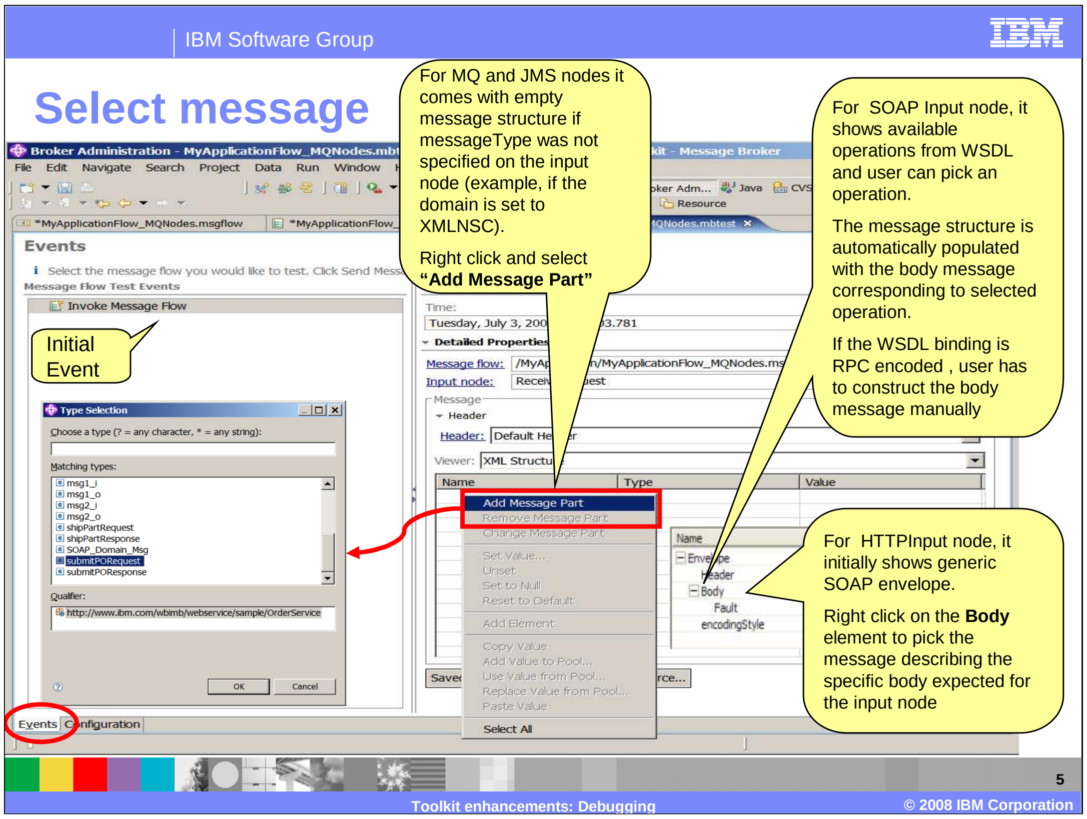
Right click on one of the Input nodes in the flow and select **Test**

Test Client Supports:

- MQInput
- SOAPInput
- HTTPInput
- JMSInput



To invoke the integrated test client, first right-click the input node of the flow you are testing. Select the “Test…” item on the pop-up list. This opens the test client dialogue.



Select message

Initial Event

For MQ and JMS nodes it comes with empty message structure if messageType was not specified on the input node (example, if the domain is set to XMLNSC).

Right click and select **"Add Message Part"**

For SOAP Input node, it shows available operations from WSDL and user can pick an operation.

The message structure is automatically populated with the body message corresponding to selected operation.

If the WSDL binding is RPC encoded , user has to construct the body message manually

For HTTPInput node, it initially shows generic SOAP envelope.

Right click on the **Body** element to pick the message describing the specific body expected for the input node

The test client window opens on the Events tab, as shown on this slide. When you right-click on the Message Body, you are presented with several options, depending on the type of input node.

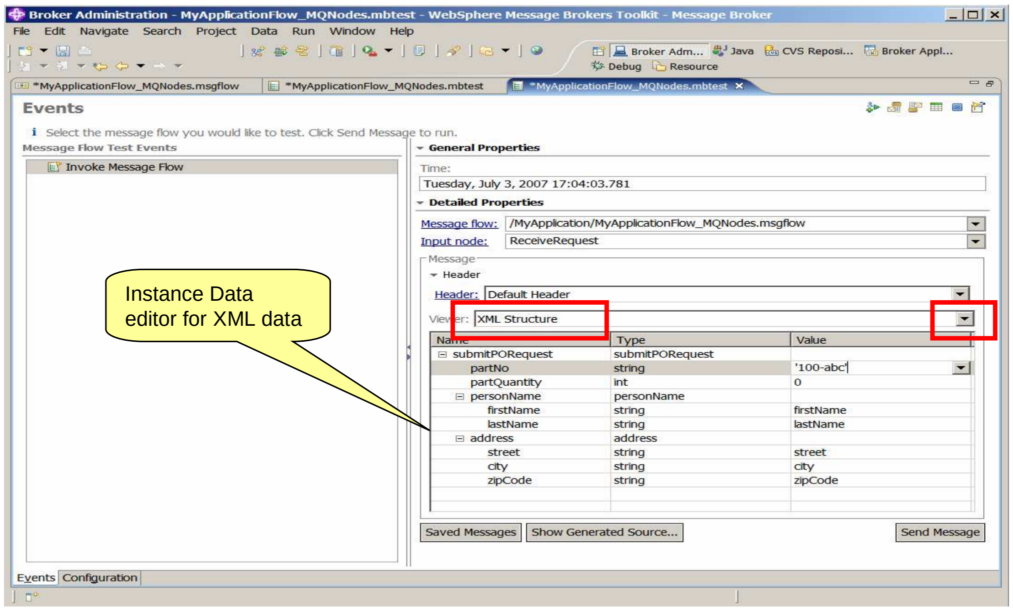
If the node is an MQ or JMS input, and if the message type was not specified on the input node, then you must select which message to input. This is done by choosing the "Add Message Part" action.

If the node is a SOAP input node, the available WSDL operations are shown, You should select the required operation from those available. The test client populates the message structure which corresponds to the chosen operation. If the WSDL uses RPC encoding, you must construct the message body manually.

If the node is an HTTP input node, the test client shows a generic SOAP envelope. To select the specific message that you require, right-click the body element.

IBM Software Group

Test client – Set input data


Broker Administration - MyApplicationFlow_MQNodes.mbtest - WebSphere Message Brokers Toolkit - Message Broker
File Edit Navigate Search Project Data Run Window Help
Broker Adm... Java CVS Reposi... Broker Appl... Debug Resource
*MyApplicationFlow_MQNodes.msgflow
*MyApplicationFlow_MQNodes.mbtest
*MyApplicationFlow_MQNodes.mbtest
Events
Message Flow Test Events
Invoke Message Flow
General Properties
Detailed Properties
Message
Header
Saved Messages
Show Generated Source...
Send Message
Events Configuration
Instance Data editor for XML data

Select the message flow you would like to test. Click Send Message to run.

Time: Tuesday, July 3, 2007 17:04:03.781

Message flow: /MyApplication/MyApplicationFlow_MQNodes.msgflow
Input node: ReceiveRequest
Header: Default Header
Viewer: XML Structure

Name	Type	Value
submitPORequest	submitPORequest	
partNo	string	'100-abc'
partQuantity	int	0
personName	personName	
firstName	string	firstName
lastName	string	lastName
address	address	
street	string	street
city	string	city
zipCode	string	zipCode

6

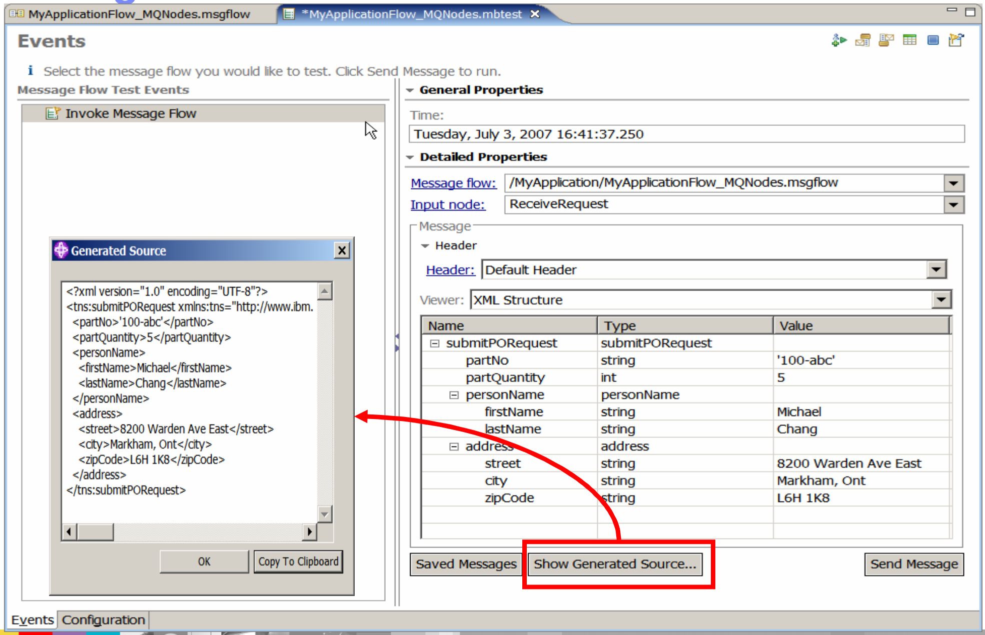
Toolkit enhancements: Debugging © 2008 IBM Corporation

The structure of the required input message is then displayed in the Detailed Properties pane, as shown on this slide. You can now provide values for each of these fields by typing directly into the value column. Text input should be contained within quotation marks.

If you have a pre-built file containing the XML data input, you can load this by clicking on the Viewer drop down, highlighted on this slide, and select Source. When you select Source, a button called “Import Source” appears at the bottom of this pane. Use this button to import the file containing the XML data; this populates the message values automatically.

IBM Software Group

Show generated source data to be sent



Time: Tuesday, July 3, 2007 16:41:37.250

Message flow: /MyApplication/MyApplicationFlow_MQNodes.msgflow

Input node: ReceiveRequest

Message

Header

Header: Default Header

Viewer: XML Structure

Name	Type	Value
submitPORequest	submitPORequest	
partNo	string	'100-abc'
partQuantity	int	5
personName	personName	
firstName	string	Michael
lastName	string	Chang
address	address	
street	string	8200 Warden Ave East
city	string	Markham, Ont
zipCode	string	L6H 1K8

Generated Source

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:submitPORequest xmlns:tns="http://www.ibm.
  <partNo>'100-abc'</partNo>
  <partQuantity>5</partQuantity>
  <personName>
    <firstName>Michael</firstName>
    <lastName>Chang</lastName>
  </personName>
  <address>
    <street>8200 Warden Ave East</street>
    <city>Markham, Ont</city>
    <zipCode>L6H 1K8</zipCode>
  </address>
</tns:submitPORequest>
```

7

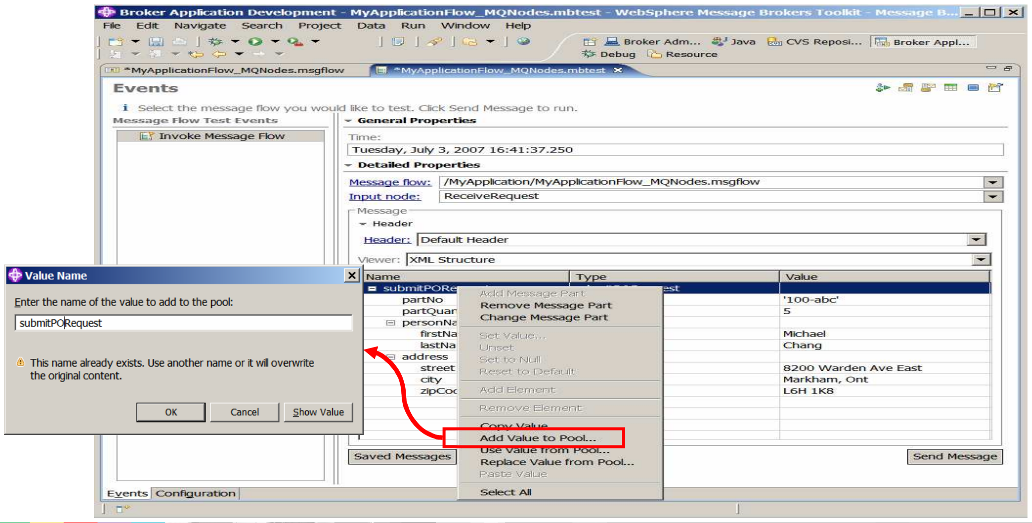
Toolkit enhancements: Debugging

© 2008 IBM Corporation

Once you have set message values as needed, you can look at the exact format of the message that is sent to the message flow, as shown on this slide.

Store value in data pool

Since you do not want to have to enter the data each time you want to test, right click and select “Add Value to Pool”


Broker Application Development - MyApplicationFlow_MQNodes.mbtest - WebSphere Message Brokers Toolkit - Message B...

Events

Select the message flow you would like to test. Click Send Message to run.

Time: Tuesday, July 3, 2007 16:41:37.250

Message flow: /MyApplication/MyApplicationFlow_MQNodes.msgflow

Input node: ReceiveRequest

Header: Default Header

Viewer: XML Structure

Name	Type	Value
submitPORe		
partNo		'100-abc'
partQuan		5
personNa		
firstNa		Michael
lastNa		Chang
address		
street		8200 Warden Ave East
city		Markham, Ont
zipCod		L6H 1K8

Value Name

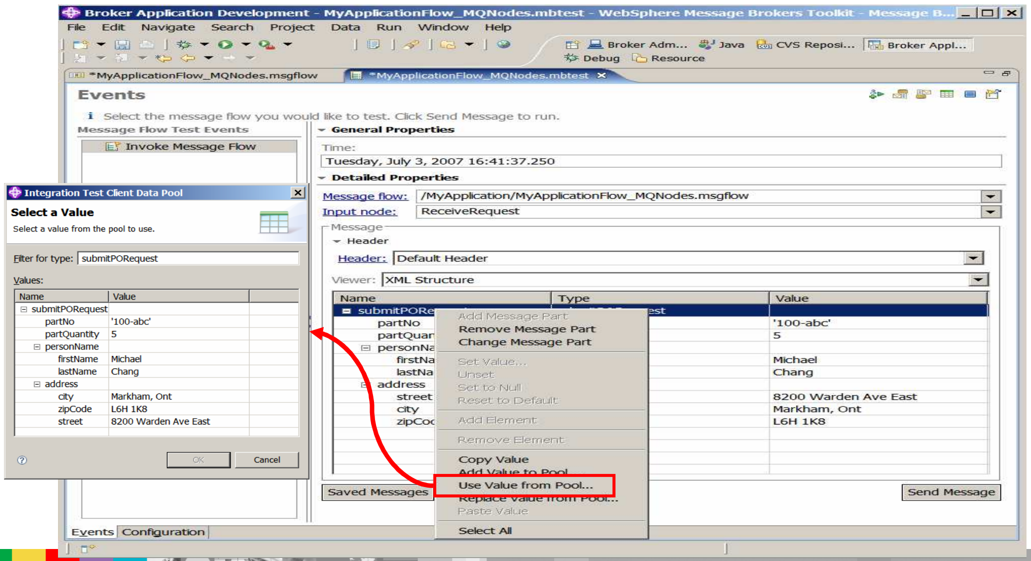
Enter the name of the value to add to the pool: submitPORequest

This name already exists. Use another name or it will overwrite the original content.

To save the values specified for this message, right-click the message name, and choose the “Add Value to Pool” action. Specify a suitable name, and the message data is stored to a save location.

Copy value from data pool

In subsequent runs, you can re-use pool values, right click and select “Use Value from Pool”


Broker Application Development - MyApplicationFlow_MQNodes.mbtest - WebSphere Message Brokers Toolkit - Message B...
File Edit Navigate Search Project Data Run Window Help
Broker Adm... Java CVS Reposi... Broker Appl... Debug Resource
*MyApplicationFlow_MQNodes.msgflow
*MyApplicationFlow_MQNodes.mbtest
Events
Message Flow Test Events
Invoke Message Flow
General Properties
Detailed Properties
Add Message Part
Remove Message Part
Change Message Part
Set Value...
Unset
Set to Null
Reset to Default
Add Element
Remove Element
Copy Value
Add Value to Pool
Use Value from Pool...
Replace Value from Pool...
Paste Value
Select All
Saved Messages
Send Message
Events Configuration
Integration Test Client Data Pool
Select a Value
OK
Cancel

Select the message flow you would like to test. Click Send Message to run.

Time: Tuesday, July 3, 2007 16:41:37.250

Message flow: /MyApplication/MyApplicationFlow_MQNodes.msgflow
Input node: ReceiveRequest
Header: Default Header
Viewer: XML Structure

Name	Type	Value
submitPORe…	…est	
partNo		'100-abc'
partQuan…		5
personNa…		
firstNa…		Michael
lastNa…		Chang
address		
street		8200 Warden Ave East
city		Markham, Ont
zipCoc…		L6H 1K8

Select a value from the pool to use.

Filter for type: submitPORequest

Values:

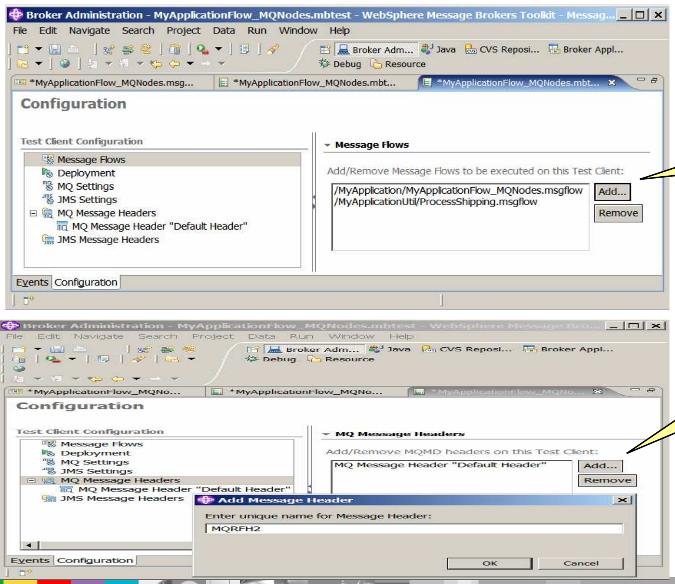
Name	Value
submitPORequest	
partNo	'100-abc'
partQuantity	5
personName	
firstName	Michael
lastName	Chang
address	
city	Markham, Ont
zipCode	L6H 1K8
street	8200 Warden Ave East

9

To retrieve this message data from the saved pool, right-click the message, and select “Use Value from Pool”. Retrieve the instance name of the saved data values.

Test client – Configuration page

- Categorized list of configuration parameters


Broker Administration - MyApplicationFlow_MQNodes.mbtest - WebSphere Message Brokers Toolkit
File Edit Navigate Search Project Data Run Window Help
Configuration
Test Client Configuration
Message Flows
Deployment
MQ Settings
JMS Settings
MQ Message Headers
MQ Message Header "Default Header"
JMS Message Headers
Add/Remove Message Flows to be executed on this Test Client:
/MyApplication/MyApplicationFlow_MQNodes.msgflow
/MyApplicationUtil/ProcessShipping.msgflow
Add...
Remove
Events Configuration
Add/Remove MQMD headers on this Test Client:
Add Message Header
Enter unique name for Message Header:
MQRFH2
OK
Cancel

Define Message flows to be included in the test run

Allows multiple message headers for MQ and JMS test messages

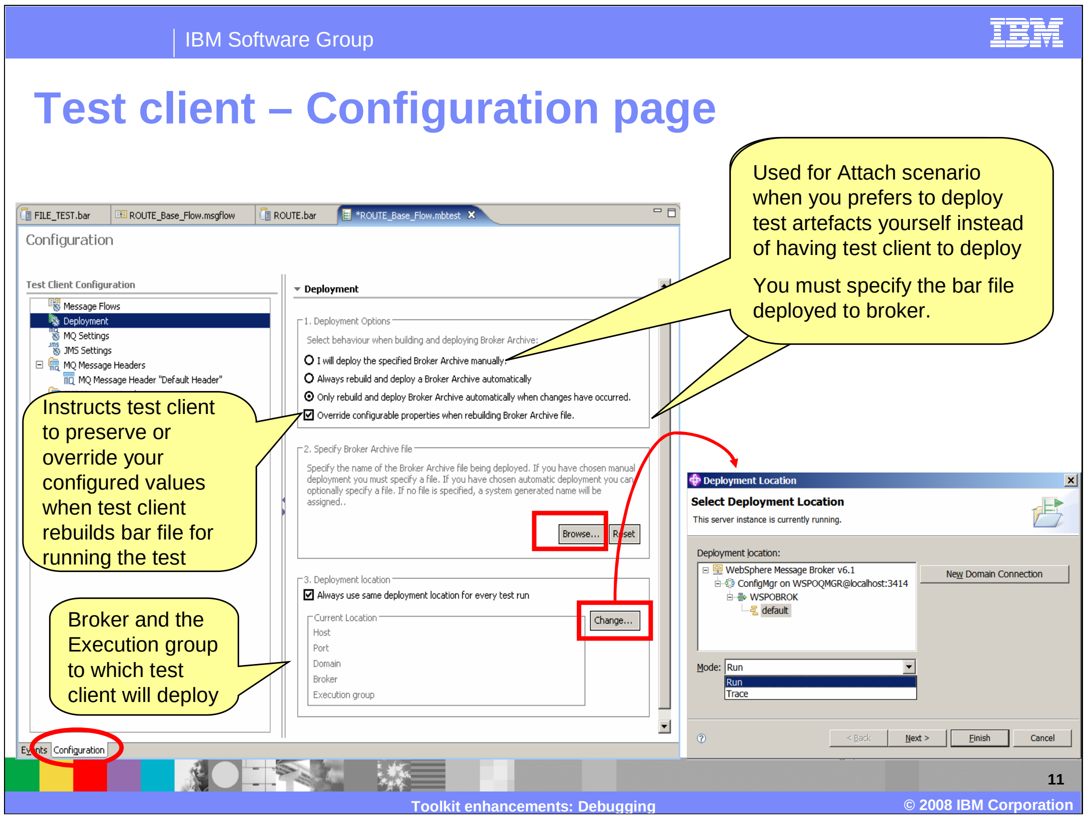
Test client composes full test message based on message header and message body

Before you run the flow with the test client, you might need to specify certain additional parameters. This is done by switching to the Configuration tab. This tab is next to the Events tab, in the bottom left corner of this pane.

If your main message flow needs additional message flows to run, you need to specify these in the test client. Click on “Message Flows” in the list of resources under the title Test Client Configuration.

In the right pane, click “Add”, which enables you to select the additional message flows that are required.

If your MQ or JMS message flow needs additional headers, click on “MQ Message Headers” in the same list. In the right pane, click “Add” to add a new header.


IBM Software Group

Test client – Configuration page

Used for Attach scenario when you prefers to deploy test artefacts yourself instead of having test client to deploy

You must specify the bar file deployed to broker.

Instructs test client to preserve or override your configured values when test client rebuilds bar file for running the test

Broker and the Execution group to which test client will deploy

FILE_TEST.bar | ROUTE_Base_Flow.msgflow | ROUTE.bar | *ROUTE_Base_Flow.mbtest

Configuration

Test Client Configuration
- Message Flows
- Deployment
- MQ Settings
- JMS Settings
- MQ Message Headers
  - MQ Message Header "Default Header"

Deployment

1. Deployment Options

Select behaviour when building and deploying Broker Archive:
- ○ I will deploy the specified Broker Archive manually
- ○ Always rebuild and deploy a Broker Archive automatically
- ◉ Only rebuild and deploy Broker Archive automatically when changes have occurred.
- ☑ Override configurable properties when rebuilding Broker Archive file.

2. Specify Broker Archive file

Specify the name of the Broker Archive file being deployed. If you have chosen manual deployment you must specify a file. If you have chosen automatic deployment you can optionally specify a file. If no file is specified, a system generated name will be assigned..

3. Deployment location

☑ Always use same deployment location for every test run

Current Location: Host, Port, Domain, Broker, Execution group

Deployment Location

Select Deployment Location

This server instance is currently running.

Deployment location:
- WebSphere Message Broker v6.1
  - ConfigMgr on WSPOQMGR@localhost:3414
    - WSPOBROK
      - default

Mode: Run

11

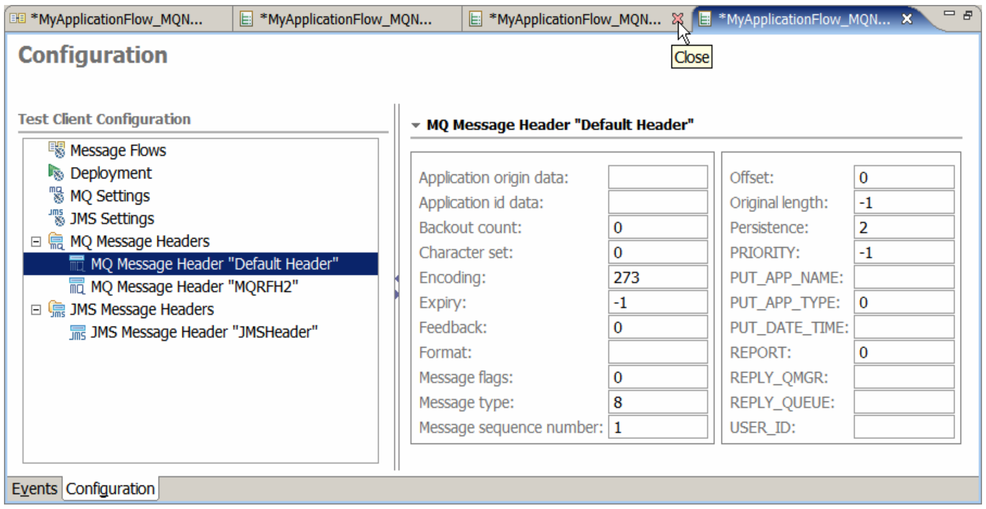
Toolkit enhancements: Debugging

© 2008 IBM Corporation

You can specify how you want to the test client to manage the deployment of the message flow that is being tested. You can choose to deploy the message flow manually, or have the test client deploy if there are changes, or have the test client deploy every time.
If you choose the “Never Deploy” option, then the bar file containing the message flow must be available. The test client asks you to specify the name of the appropriate bar file. This is shown highlighted in the pane called “Specify Broker Archive file”.
If you choose the second option, the test client re-creates a bar file with the refreshed message flow, and deploys this bar file to the broker.
If you choose the third option, the bar file is rebuilt and redeployed every time the flow is run.

For the option "Override configurable properties when rebuilding bar file," you can manually open BAR editor to configure some properties for the message flows being tested. This option instructs the test client whether those user-configured values are preserved or not when the test client rebuilds the bar file when running tests.

Test client – Set message headers


*MyApplicationFlow_MQN...
*MyApplicationFlow_MQN...
*MyApplicationFlow_MQN...
*MyApplicationFlow_MQN...
Close
Events
Configuration

Configuration

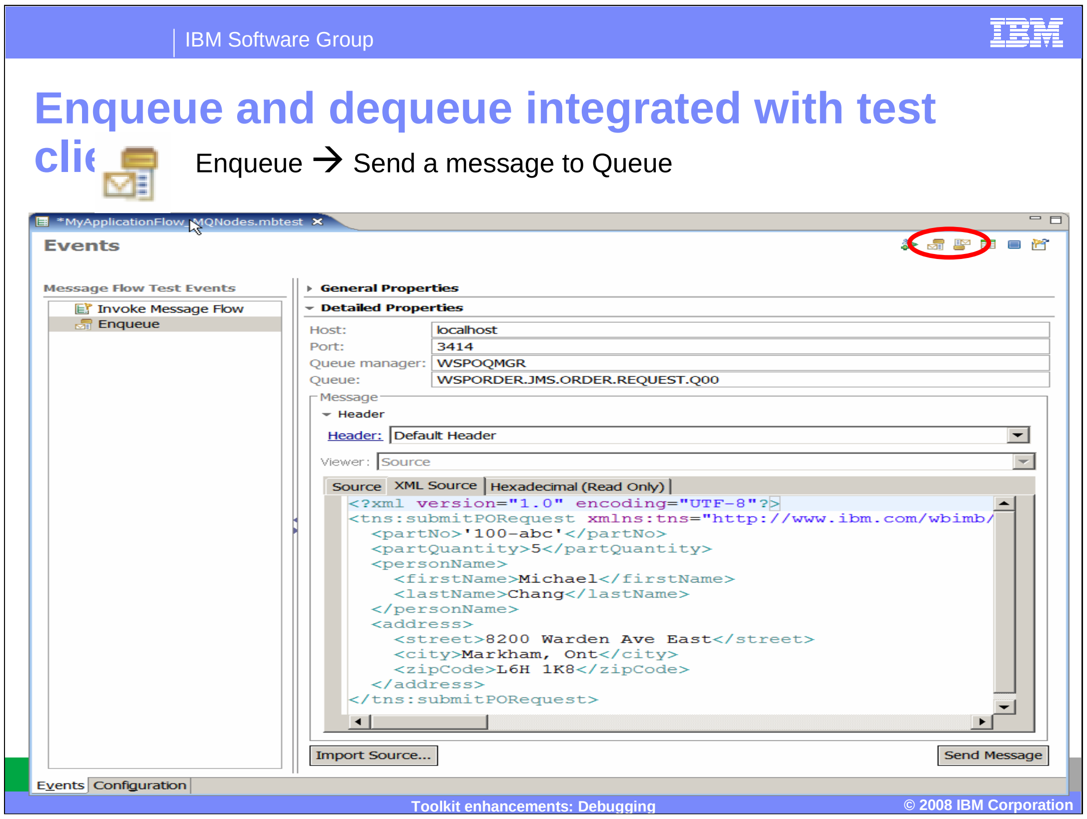
Test Client Configuration

- Message Flows
- Deployment
- MQ Settings
- JMS Settings
- MQ Message Headers
  - MQ Message Header "Default Header"
  - MQ Message Header "MQRFH2"
- JMS Message Headers
  - JMS Message Header "JMSHeader"

MQ Message Header "Default Header"

Application origin data:		Offset:	0
Application id data:		Original length:	-1
Backout count:	0	Persistence:	2
Character set:	0	PRIORITY:	-1
Encoding:	273	PUT_APP_NAME:	
Expiry:	-1	PUT_APP_TYPE:	0
Feedback:	0	PUT_DATE_TIME:	
Format:		REPORT:	0
Message flags:	0	REPLY_QMGR:	
Message type:	8	REPLY_QUEUE:	
Message sequence number:	1	USER_ID:	

As mentioned earlier, you can specify the values for certain message headers. You can also add optional headers. Clicking on the selected header allows you to specify values for each of the fields within the header.



IBM Software Group

Enqueue and dequeue integrated with test clie

Enqueue → Send a message to Queue

*MyApplicationFlow_MQNodes.mbtest

Events

Message Flow Test Events
- Invoke Message Flow
- Enqueue

General Properties

Detailed Properties

Host:	localhost
Port:	3414
Queue manager:	WSPOQMGR
Queue:	WSPORDER.JMS.ORDER.REQUEST.Q00

Message

Header

Header: Default Header

Viewer: Source

Source | XML Source | Hexadecimal (Read Only)

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:submitPORequest xmlns:tns="http://www.ibm.com/wbimb/
  <partNo>'100-abc'</partNo>
  <partQuantity>5</partQuantity>
  <personName>
    <firstName>Michael</firstName>
    <lastName>Chang</lastName>
  </personName>
  <address>
    <street>8200 Warden Ave East</street>
    <city>Markham, Ont</city>
    <zipCode>L6H 1K8</zipCode>
  </address>
</tns:submitPORequest>
```

Import Source... | Send Message

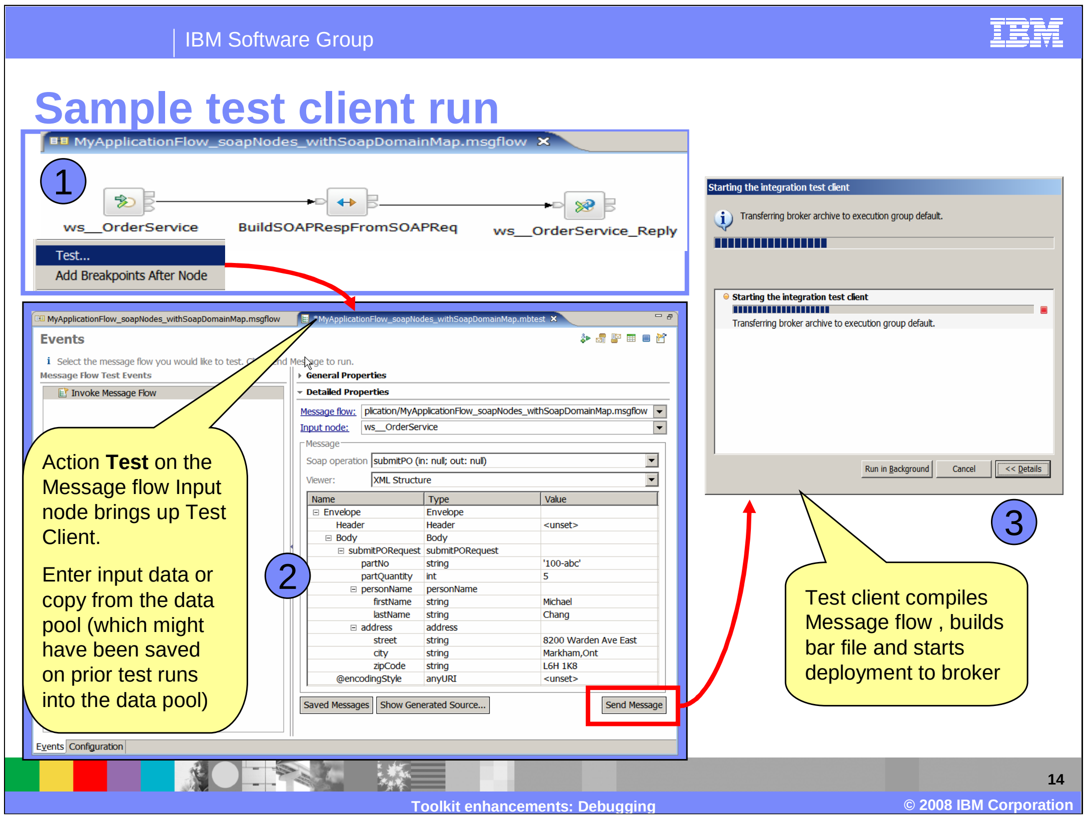
Events | Configuration

Toolkit enhancements: Debugging © 2008 IBM Corporation

The enqueue/dequeue function has been fully integrated into the Test Client in version 6.1. This function is invoked by clicking the icons in the top right corner of the Events pane.

When you click the Enqueue icon, the display changes to that shown on this slide. You can then send a message to a particular queue using this screen. The data payload is populated using the test client functions shown earlier.

To return to the normal test client display, click on the Invoke icon, which is just to the left of the Enqueue and Dequeue icons.



Sample test client run

MyApplicationFlow_soapNodes_withSoapDomainMap.msgflow
1
ws__OrderService
BuildSOAPRespFromSOAPReq
ws__OrderService_Reply
Test...
Add Breakpoints After Node
2
3

*MyApplicationFlow_soapNodes_withSoapDomainMap.mbtest

Events

Select the message flow you would like to test. Click Send Message to run.

Message Flow Test Events

Invoke Message Flow

General Properties

Detailed Properties

Message flow: plication/MyApplicationFlow_soapNodes_withSoapDomainMap.msgflow

Input node: ws__OrderService

Message

Soap operation: submitPO (in: null; out: null)

Viewer: XML Structure

Name	Type	Value
Envelope	Envelope	
Header	Header	<unset>
Body	Body	
submitPORequest	submitPORequest	
partNo	string	'100-abc'
partQuantity	int	5
personName	personName	
firstName	string	Michael
lastName	string	Chang
address	address	
street	string	8200 Warden Ave East
city	string	Markham,Ont
zipCode	string	L6H 1K8
@encodingStyle	anyURI	<unset>

Starting the integration test client

Transferring broker archive to execution group default.

Starting the integration test client

Transferring broker archive to execution group default.

Action **Test** on the Message flow Input node brings up Test Client.

Enter input data or copy from the data pool (which might have been saved on prior test runs into the data pool)

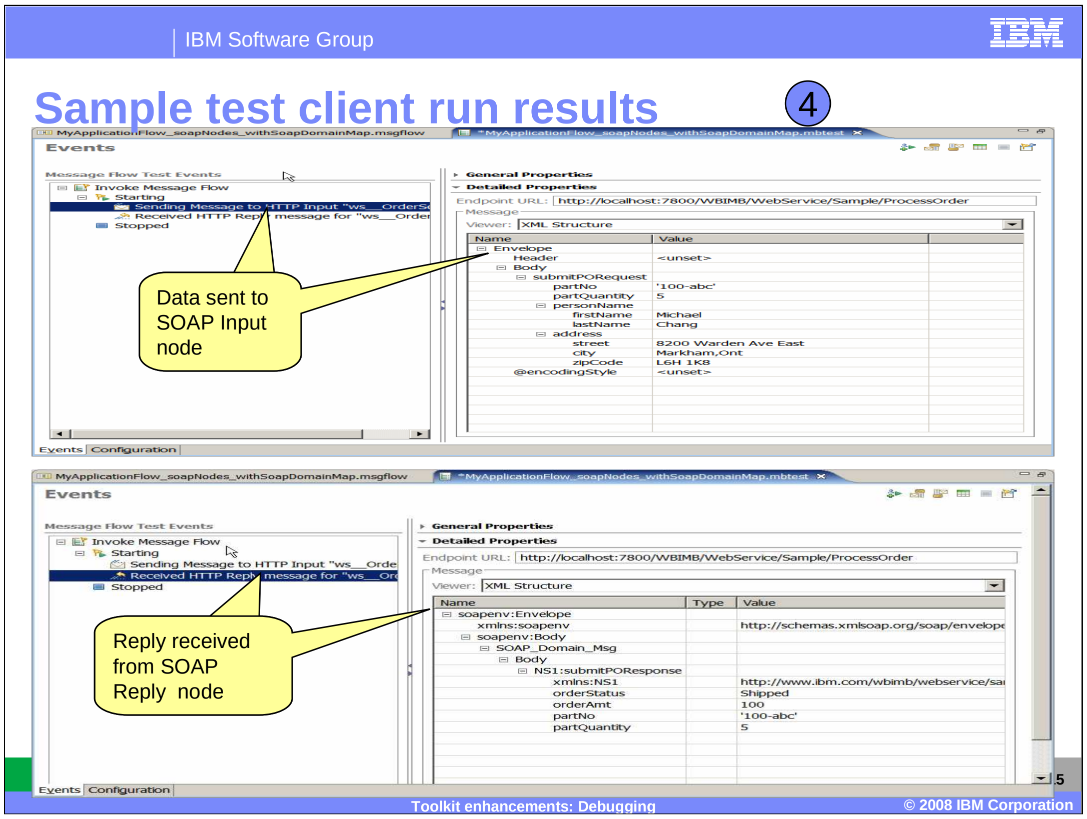
Test client compiles Message flow , builds bar file and starts deployment to broker

This slide shows a sample run of the test client.

This example shows a simple message flow. It has a SOAP input node, a mapping node and a SOAP reply. Stage 1 is to right-click on the SOAP input node and select “Test…” to open the test client window as shown.

Because this is a SOAP node, the WSDL and operation that are required are already available, so this information is already populated into the test client. Alternatively, this data can be manually entered, or retrieved from the data pool, shown as stage 2.

When you click “Send Message”, the test client creates the bar file, and deploys it to the runtime execution group, shown as stage 3 on this slide.



IBM Software Group

Sample test client run results

4

MyApplicationFlow_soapNodes_withSoapDomainMap.msgflow | *MyApplicationFlow_soapNodes_withSoapDomainMap.mbtest

Events

Message Flow Test Events

- Invoke Message Flow
  - Starting
    - Sending Message to HTTP Input "ws__OrderS
    - Received HTTP Reply message for "ws__Order
  - Stopped

General Properties

Detailed Properties

Endpoint URL: http://localhost:7800/WBIMB/WebService/Sample/ProcessOrder

Message

Viewer: XML Structure

Name	Value
Envelope	
Header	<unset>
Body	
submitPORequest	
partNo	'100-abc'
partQuantity	5
personName	
firstName	Michael
lastName	Chang
address	
street	8200 Warden Ave East
city	Markham,Ont
zipCode	L6H 1K8
@encodingStyle	<unset>

Events | Configuration

MyApplicationFlow_soapNodes_withSoapDomainMap.msgflow | *MyApplicationFlow_soapNodes_withSoapDomainMap.mbtest

Events

Message Flow Test Events

- Invoke Message Flow
  - Starting
    - Sending Message to HTTP Input "ws__Orde
    - Received HTTP Reply message for "ws__Or
  - Stopped

General Properties

Detailed Properties

Endpoint URL: http://localhost:7800/WBIMB/WebService/Sample/ProcessOrder

Message

Viewer: XML Structure

Name	Type	Value
soapenv:Envelope		
xmlns:soapenv		http://schemas.xmlsoap.org/soap/envelope
soapenv:Body		
SOAP_Domain_Msg		
Body		
NS1:submitPOResponse		
xmlns:NS1		http://www.ibm.com/wbimb/webservice/sa
orderStatus		Shipped
orderAmt		100
partNo		'100-abc'
partQuantity		5

Events | Configuration

This slide shows the execution of the message flow. The first screen capture shows the input HTTP request, with the payload of the HTTP message in the right pane.

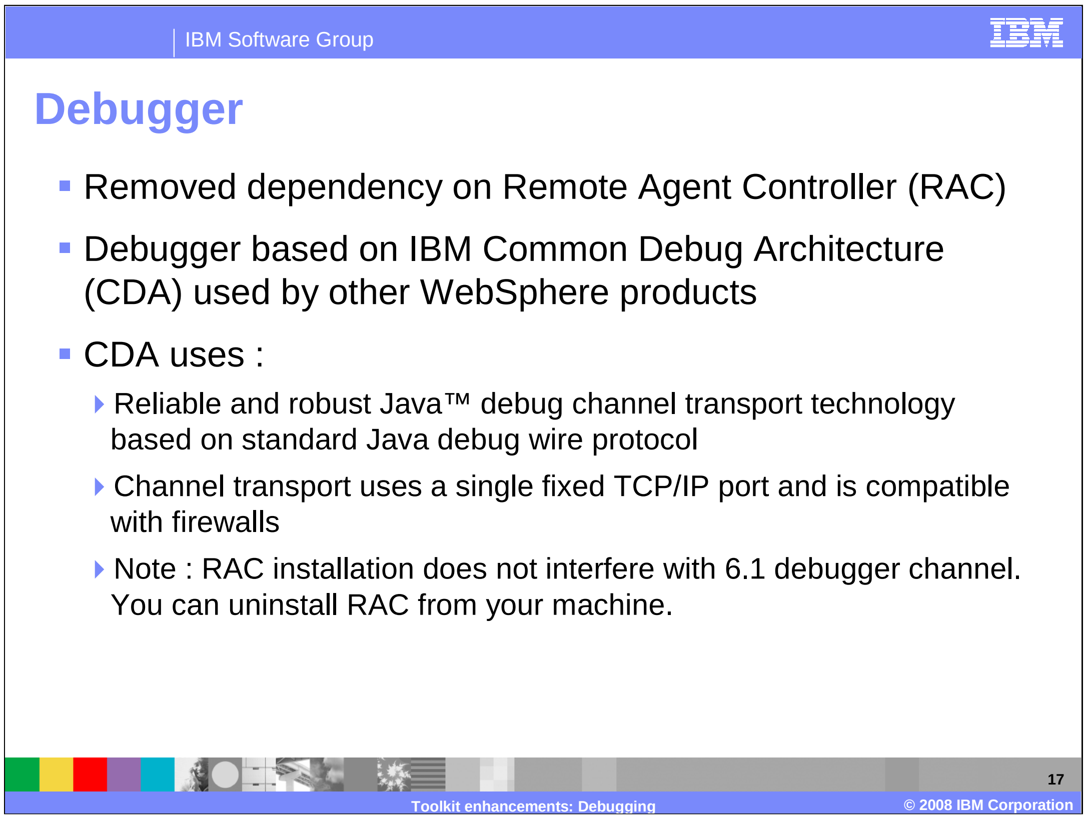
The second screen capture shows the HTTP response highlighted, with the payload of the response again shown in the right pane.

As the message flow is re-run, each input and output message is recorded in the test client. When you exit from the test client, you can save this record as a separate file, with the extension “.mbtest”.

Section

Debug

This section covers the debug function in version 6.1

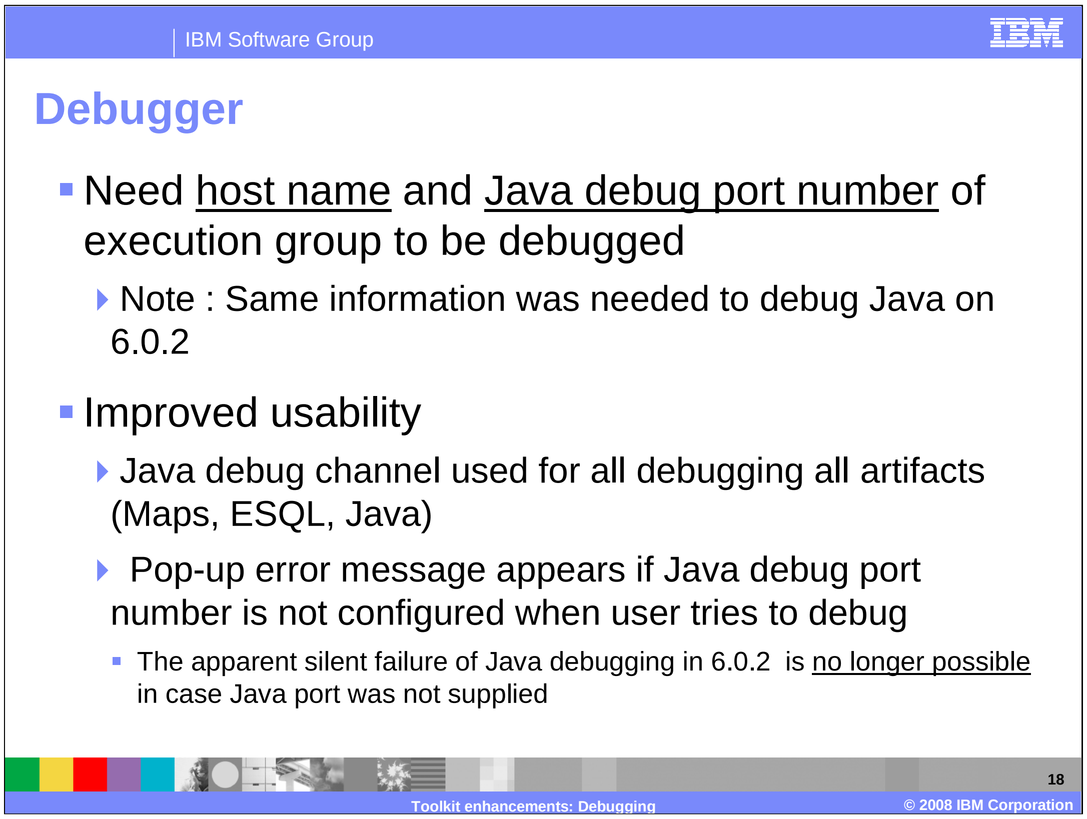


Debugger

- Removed dependency on Remote Agent Controller (RAC)
- Debugger based on IBM Common Debug Architecture (CDA) used by other WebSphere products
- CDA uses :
  - Reliable and robust Java™ debug channel transport technology based on standard Java debug wire protocol
  - Channel transport uses a single fixed TCP/IP port and is compatible with firewalls
  - Note : RAC installation does not interfere with 6.1 debugger channel. You can uninstall RAC from your machine.

In version 6.1, the debugger has removed the dependency on Rational Agent Controller. This has been replaced by the Java Debug Channel, and this makes the configuration of the debug function easier than in previous versions. This needs the configuration of a single TCP/IP port on the runtime component. If the broker runtime is separated from the Toolkit by a firewall, then the firewall should be configured to permit this port.

This new facility is fully integrated into Message Broker 6.1, and you do not have to install a separate product or component. However, if you have an existing installation of Rational Agent Controller, this does not interfere with version 6.1. However, you may want to uninstall this when the earlier broker installation is removed.



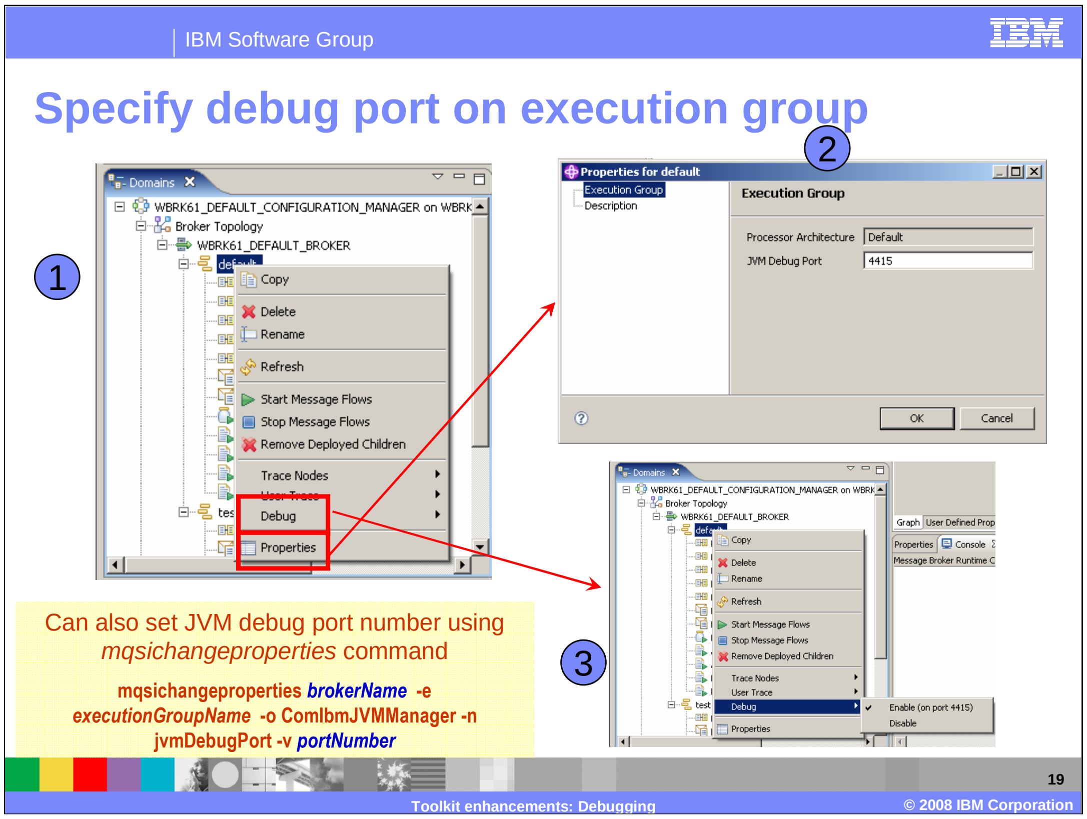
Debugger

- Need host name and Java debug port number of execution group to be debugged
  - Note : Same information was needed to debug Java on 6.0.2
- Improved usability
  - Java debug channel used for all debugging all artifacts (Maps, ESQL, Java)
  - Pop-up error message appears if Java debug port number is not configured when user tries to debug
    - The apparent silent failure of Java debugging in 6.0.2 is no longer possible in case Java port was not supplied

The debugger is invoked using the same information as in earlier releases, namely the host name of the broker runtime, and the port that has been specified to run the Java debugger.

The same debug channel is used to debug all components in a message flow, whereas in earlier versions this was inconsistent. Version 6.1 therefore provides a more consistent user interface to the debug function.

Hence, in version 6.1, you must configure a Java debug port. Once this has been done, all broker components can be debugged in the same debug session, and against the same Java debug channel.



Specify debug port on execution group

Execution Group

Processor Architecture: Default

JVM Debug Port: 4415

Can also set JVM debug port number using *mqsichangeproperties* command

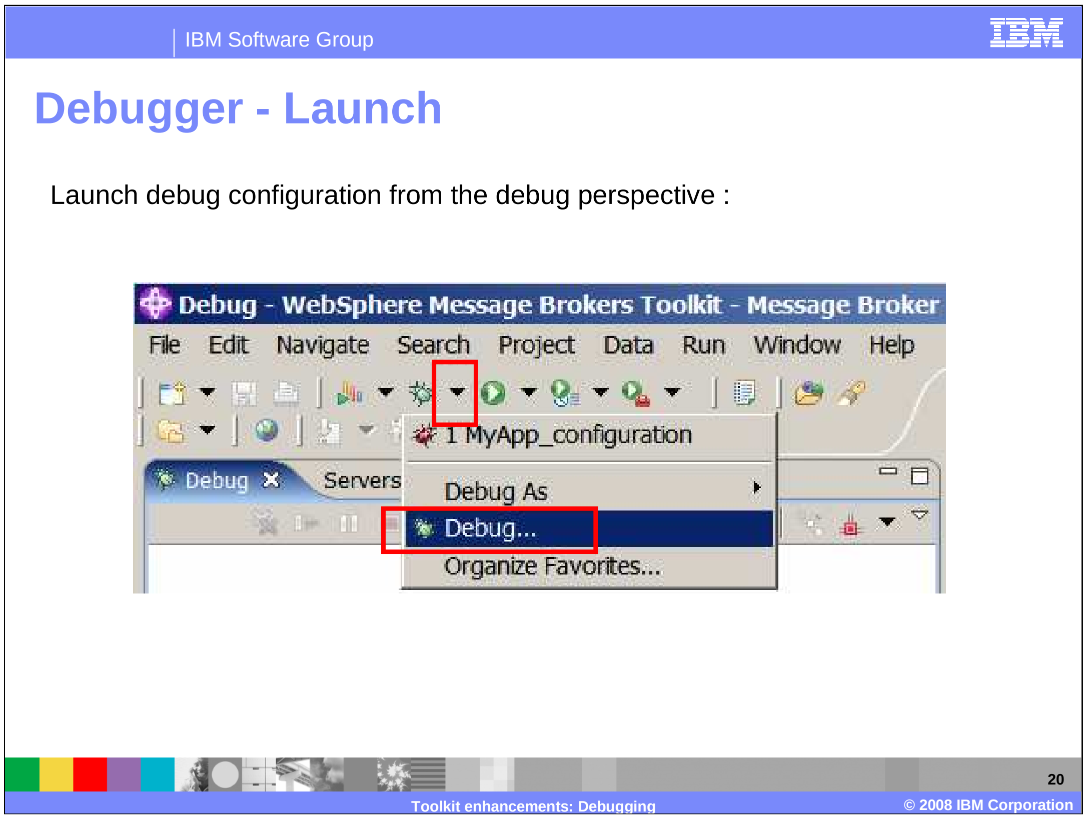
mqsichangeproperties *brokerName* -e *executionGroupName* -o ComIbmJVMManager -n jvmDebugPort -v *portNumber*

Before using the debugger, you must configure the required port on the broker's execution group. You can do this using the Administration perspective.

First, in task 1 on this slide, right-click on the execution group, and select the Properties action on the pop-up. On the resulting window, specify the port number. This example shows 4415. Click on OK. If this is the first time that this port number has been used, you must restart the execution group. This can be done from this same perspective by stopping all message flows, and then restarting all message flows.

Second, to check that the required port has been enabled, right-click the execution group, and select the Debug action. This opens the window shown as task 3. It should show that the debugger is enabled, and is active on the specified port.

This port can also be specified using the mqsichangeproperties command.



IBM Software Group

Debugger - Launch

Launch debug configuration from the debug perspective :

Debug - WebSphere Message Brokers Toolkit - Message Broker
File Edit Navigate Search Project Data Run Window Help
1 MyApp_configuration
Debug As
Debug...
Organize Favorites...
Debug
Servers

20

Toolkit enhancements: Debugging

© 2008 IBM Corporation

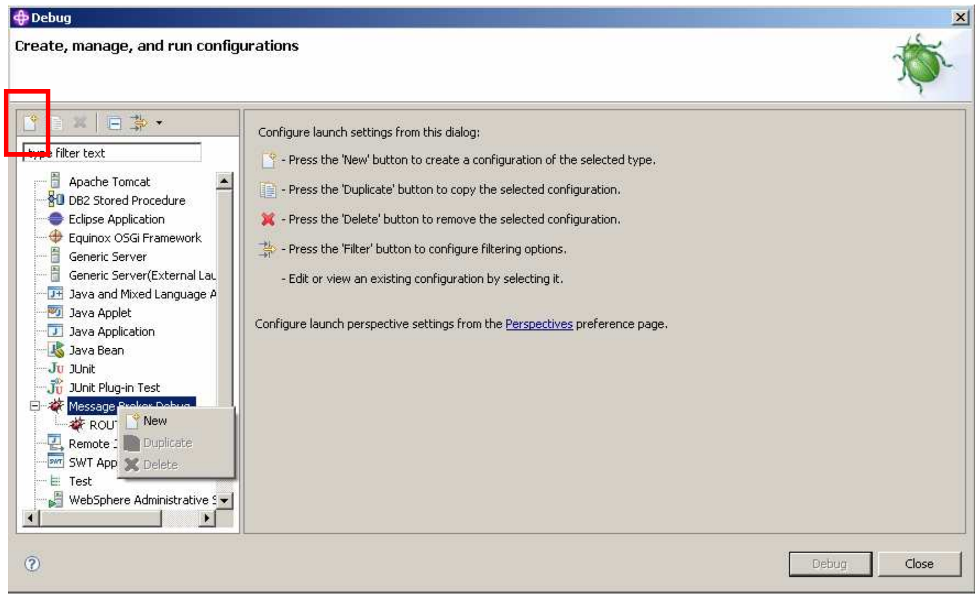
This slide shows how to start the debug function from the Toolkit.

First, open the Debug perspective, then click on the down-arrow next to the debug icon. This is shown highlighted on this slide.

Second, select Debug from the action list, again shown highlighted on the slide.

This opens the Debug control window.

Debugger – Create new configuration



Create, manage, and run configurations

Configure launch settings from this dialog:

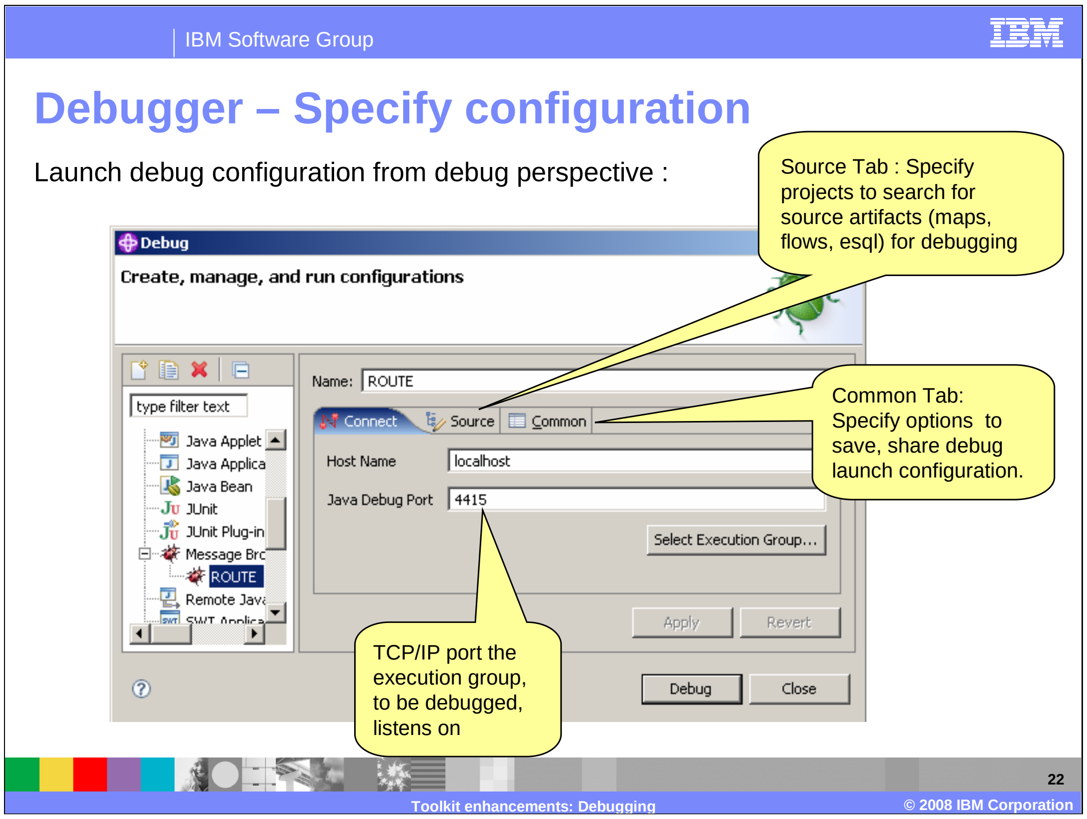
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Click on “Message Broker Debug” in the pane on the left side. To create a new debug configuration, right-click “Message Broker Debug”, then select “New”.

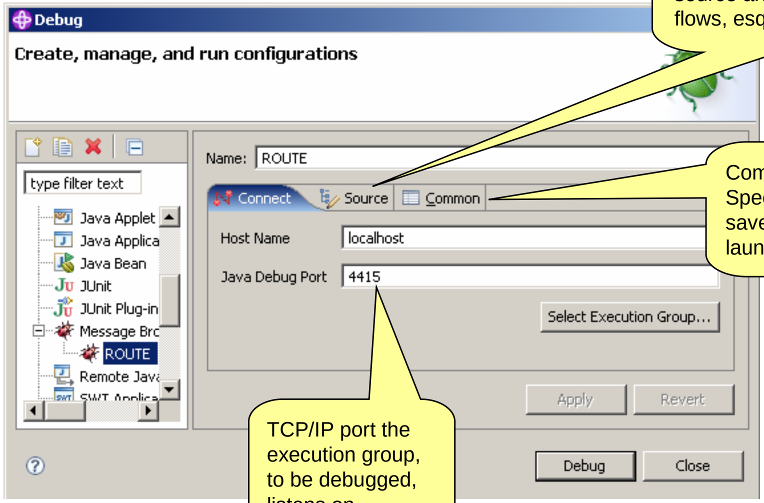
Alternatively, you can select “New” from the control icons near the top of this window, shown as highlighted.

If you want to update the details of an existing Debug configuration, highlight the required configuration; and the details are shown in the right pane.



Debugger – Specify configuration

Launch debug configuration from debug perspective :


Debug
Create, manage, and run configurations
type filter text
Java Applet
Java Applica
Java Bean
JUnit
JUnit Plug-in
Message Bro
ROUTE
Remote Java
SWT Applica
Connect
Source
Common
Select Execution Group...
Apply
Revert
Debug
Close
Source Tab : Specify projects to search for source artifacts (maps, flows, esql) for debugging
Common Tab: Specify options to save, share debug launch configuration.
TCP/IP port the execution group, to be debugged, listens on

Name: ROUTE

Host Name: localhost

Java Debug Port: 4415

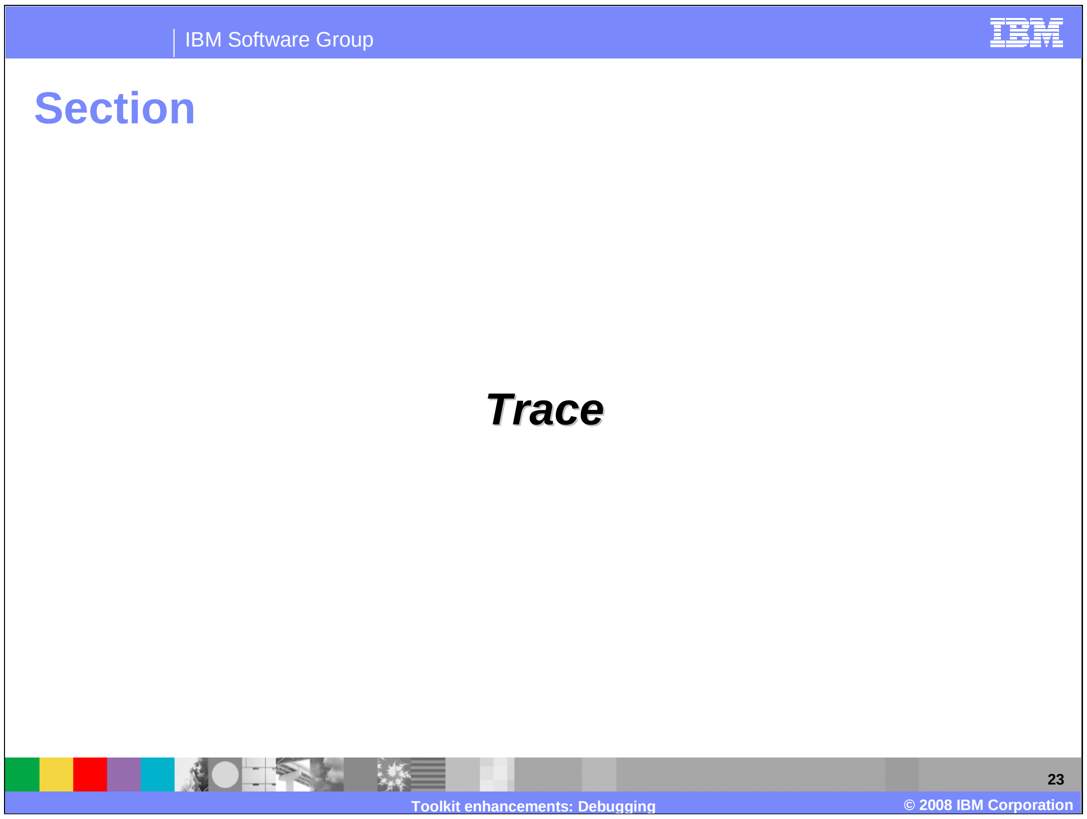
The debug control window is now used to specify the required Java debug port, and the source code of the message flow that is going to be debugged.

First, if this is a new configuration, specify the name of the Java debug configuration that you are going to create.

Second, specify the Java debug port. If you already know the port you are using, this can be done manually by typing into the field. Alternatively, you can click “Select Execution Group”, which then queries the broker for all available execution groups. From the returned list, select the required port.

Finally, on the Source tab, specify the Toolkit location of the message flow that you are debugging.

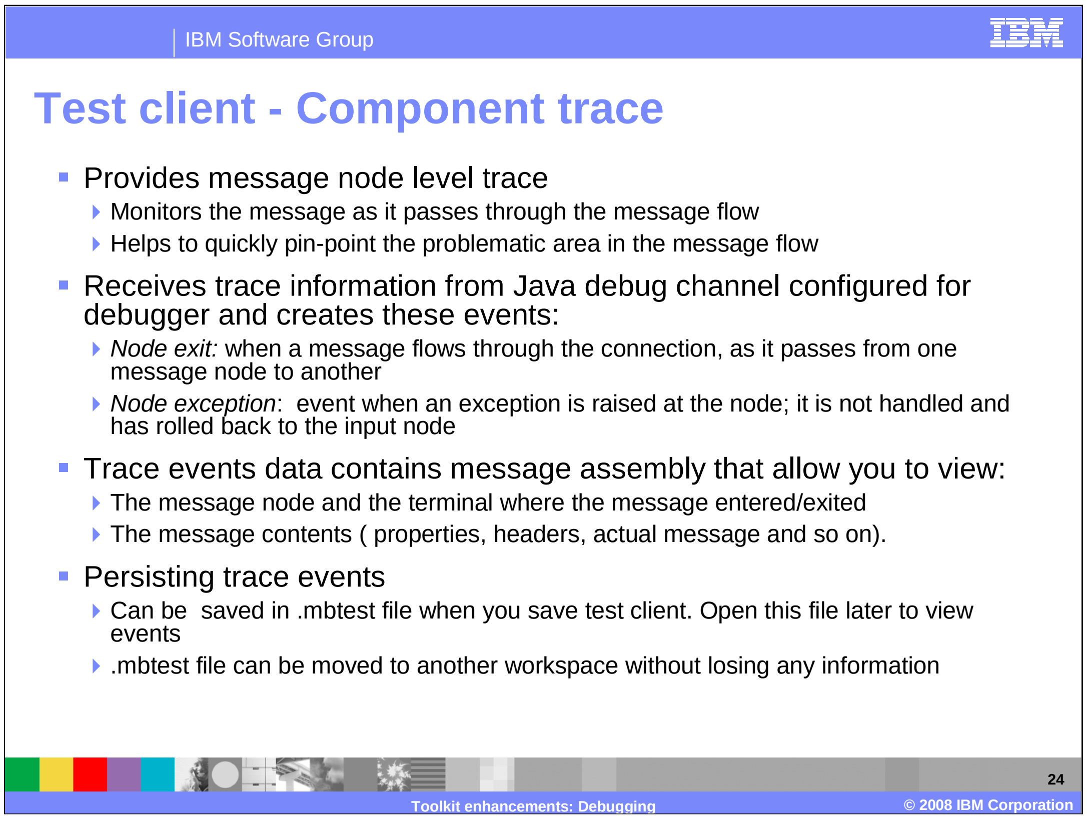
The Common tab is optional, and is used to save local information to enable sharing of the debug configuration.



Section

Trace

This final section covers the trace tools introduced in version 6.1.

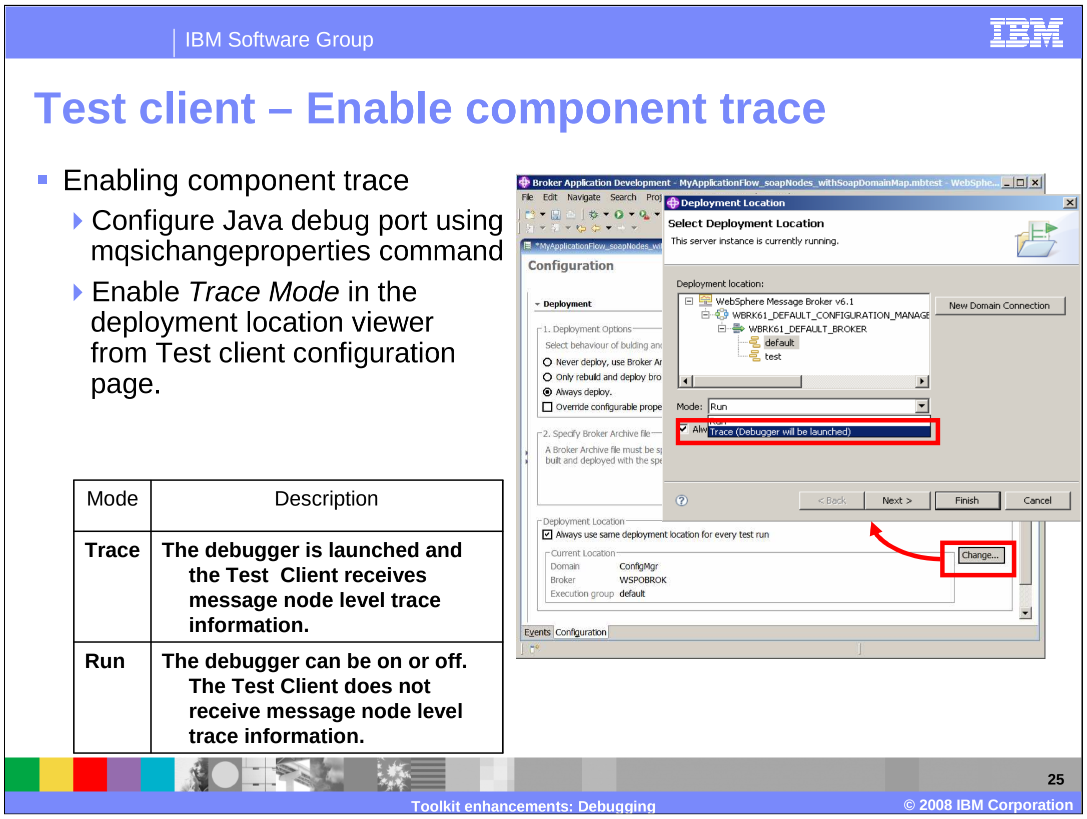


Test client - Component trace

- Provides message node level trace
  - Monitors the message as it passes through the message flow
  - Helps to quickly pin-point the problematic area in the message flow
- Receives trace information from Java debug channel configured for debugger and creates these events:
  - *Node exit:* when a message flows through the connection, as it passes from one message node to another
  - *Node exception*: event when an exception is raised at the node; it is not handled and has rolled back to the input node
- Trace events data contains message assembly that allow you to view:
  - The message node and the terminal where the message entered/exited
  - The message contents (properties, headers, actual message and so on).
- Persisting trace events
  - Can be saved in .mbtest file when you save test client. Open this file later to view events
  - .mbtest file can be moved to another workspace without losing any information

The test client component trace function traces a message flow as it runs, and records the output from that trace into the Test Client. It shows the path that the message flow has taken, and records all debug information generated throughout the flow. The two key trace points are “Node exit” and “Node exception”. If an exception is raised, but not handled by the message flow, the exception data is caught and stored in the test client.

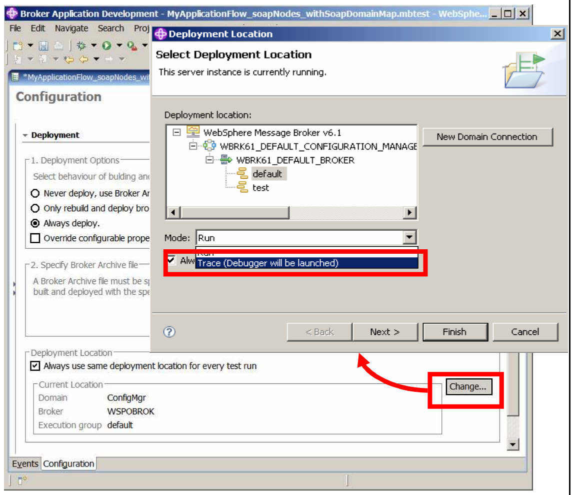
The trace tool collects the message properties, message headers, and contents of the message body, as it passes through the flow, from node to node. As with other aspects of the test client, this can be saved in the “.mbtest” file.



Test client – Enable component trace

- Enabling component trace
  - Configure Java debug port using mqsichangeproperties command
  - Enable *Trace Mode* in the deployment location viewer from Test client configuration page.

Mode	Description
Trace	**The debugger is launched and the Test Client receives message node level trace information.**
Run	**The debugger can be on or off. The Test Client does not receive message node level trace information.**



Broker Application Development - MyApplicationFlow_soapNodes_withSoapDomainMap.mbtest - WebSphe...
File Edit Navigate Search Proj
Configuration
Deployment
Deployment Location
Select Deployment Location
This server instance is currently running.
Deployment location:
WebSphere Message Broker v6.1
WBRK61_DEFAULT_CONFIGURATION_MANAGE
WBRK61_DEFAULT_BROKER
default
test
New Domain Connection
Mode: Run
Trace (Debugger will be launched)
< Back
Next >
Finish
Cancel
1. Deployment Options
Select behaviour of building an
Never deploy, use Broker Ar
Only rebuild and deploy bro
Always deploy.
Override configurable prope
2. Specify Broker Archive file
A Broker Archive file must be s
built and deployed with the spe
Deployment Location
Always use same deployment location for every test run
Current Location
Change...
Events
Configuration

Domain ConfigMgr
Broker WSPOBROK
Execution group default

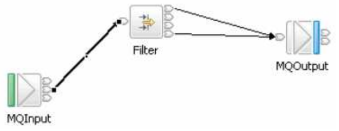
This slide shows how to enable the Component Trace.

On the Configuration tab of the Test Client, click “Change” in the Deployment Location pane. On the resulting window, change the Mode to “Trace”.

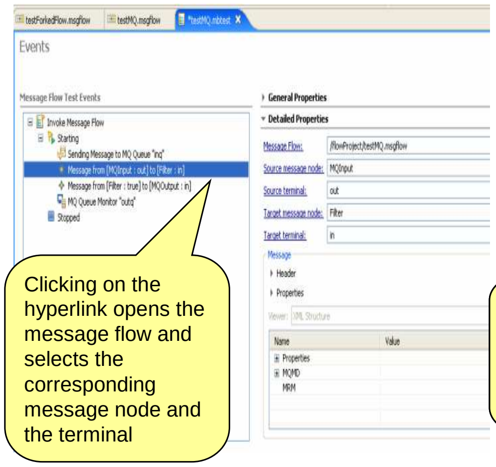
When this is done, the mode is set to trace, the debugger is launched, and the test client starts.

Sample component trace for message flow

Sample message flow



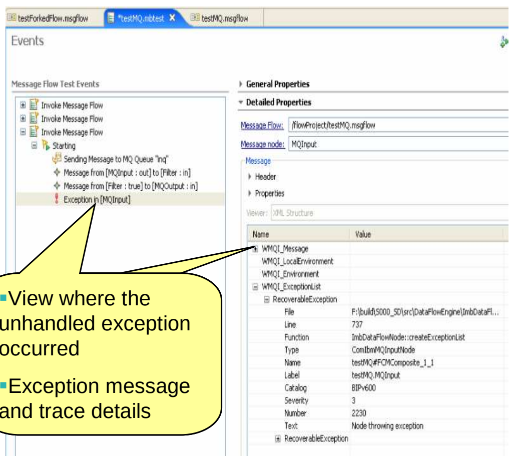
Successful test



```
testForkedFlow.msgflow   testMQ.msgflow   *testMQ.mbtest X
Events

Message Flow Test Events
⊟ Invoke Message Flow
   ⊟ Starting
        Sending Message to MQ Queue "inq"
        Message from [MQInput : out] to [Filter : in]
        Message from [Filter : true] to [MQOutput : in]
        MQ Queue Monitor "outq"
     Stopped

General Properties
Detailed Properties
Message Flow:          /flowProject/testMQ.msgflow
Source message node:   MQInput
Source terminal:       out
Target message node:   Filter
Target terminal:       in
Message
  Header
  Properties
Viewer: XML Structure
Name            Value
⊞ Properties
⊞ MQMD
  MRM
```

Clicking on the hyperlink opens the message flow and selects the corresponding message node and the terminal

Typical failed test



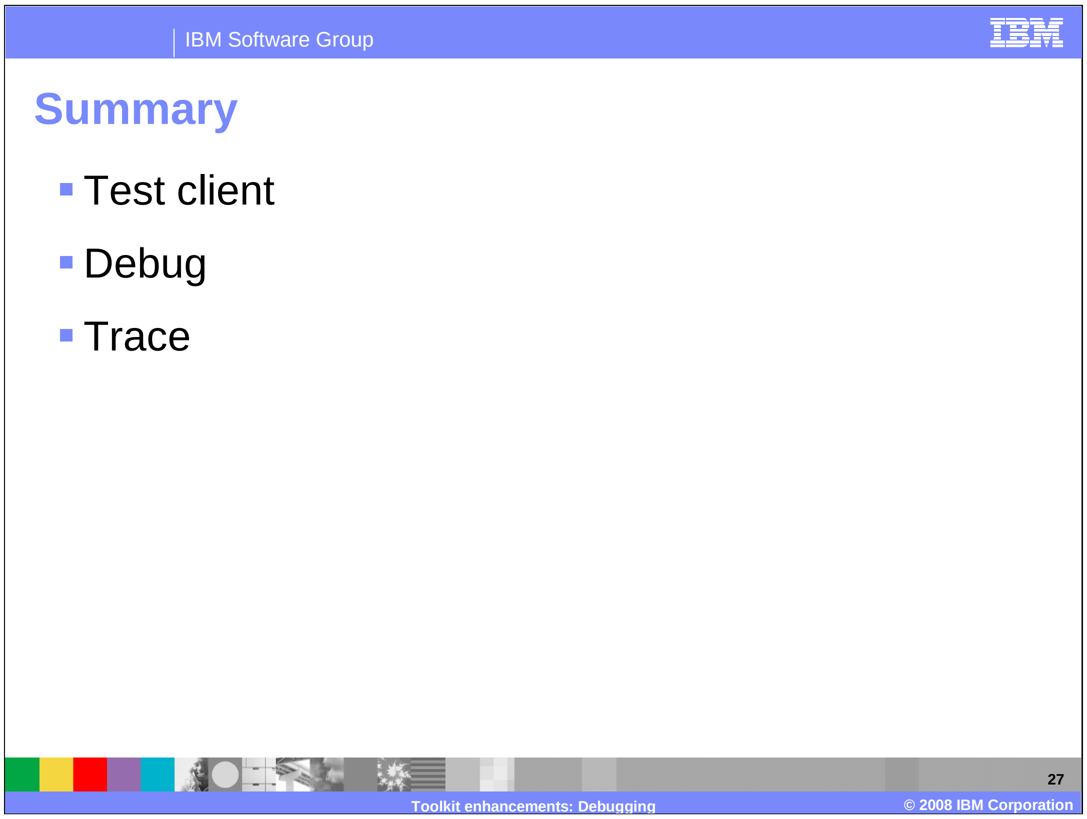
```
testForkedFlow.msgflow   *testMQ.mbtest X   testMQ.msgflow
Events

Message Flow Test Events
⊞ Invoke Message Flow
⊞ Invoke Message Flow
⊟ Invoke Message Flow
   ⊟ Starting
        Sending Message to MQ Queue "inq"
        Message from [MQInput : out] to [Filter : in]
        Message from [Filter : true] to [MQOutput : in]
        Exception in [MQInput]

General Properties
Detailed Properties
Message Flow:   /flowProject/testMQ.msgflow
Message node:   MQInput
Message
  Header
  Properties
Viewer: XML Structure
Name                        Value
⊞ WMQI_Message
  WMQI_LocalEnvironment
  WMQI_Environment
⊟ WMQI_ExceptionList
   ⊟ RecoverableException
        File                F:\build\S000_SD\src\DataFlowEngine\ImbDataFl...
        Line                737
        Function            ImbDataFlowNode::createExceptionList
        Type                ComIbmMQInputNode
        Name                testMQ#FCMComposite_1_1
        Label               testMQ.MQInput
        Catalog             BIPv600
        Severity            3
        Number              2230
        Text                Node throwing exception
      ⊞ RecoverableException
```

- View where the unhandled exception occurred
- Exception message and trace details

This slide shows an example of the trace tool. This example has an MQ input node, a filter node and an MQ output node.

The screen capture on the left shows a successful flow execution. Each line indicates the flow has moved from one node to the next. Highlighting each node enables the message data to be displayed in the right pane.

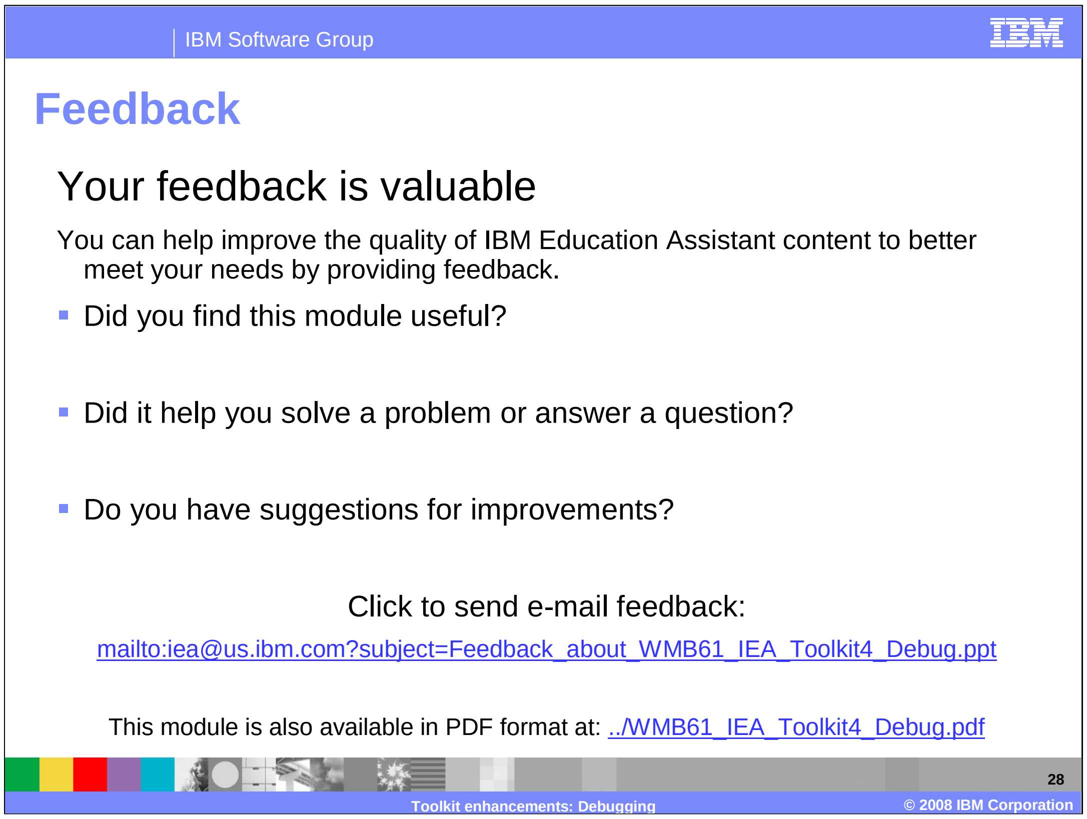
The screen capture on the right shows a failed message flow. As with the successful flow, all message data is available to be viewed. In addition, at the point of failure, the exception message is available in the test client. This information is the same as is available in a user trace file.



Summary

- Test client
- Debug
- Trace

In summary, this presentation has covered improvements made in the area of application testing and debugging.


IBM Software Group

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_Toolkit4_Debug.ppt

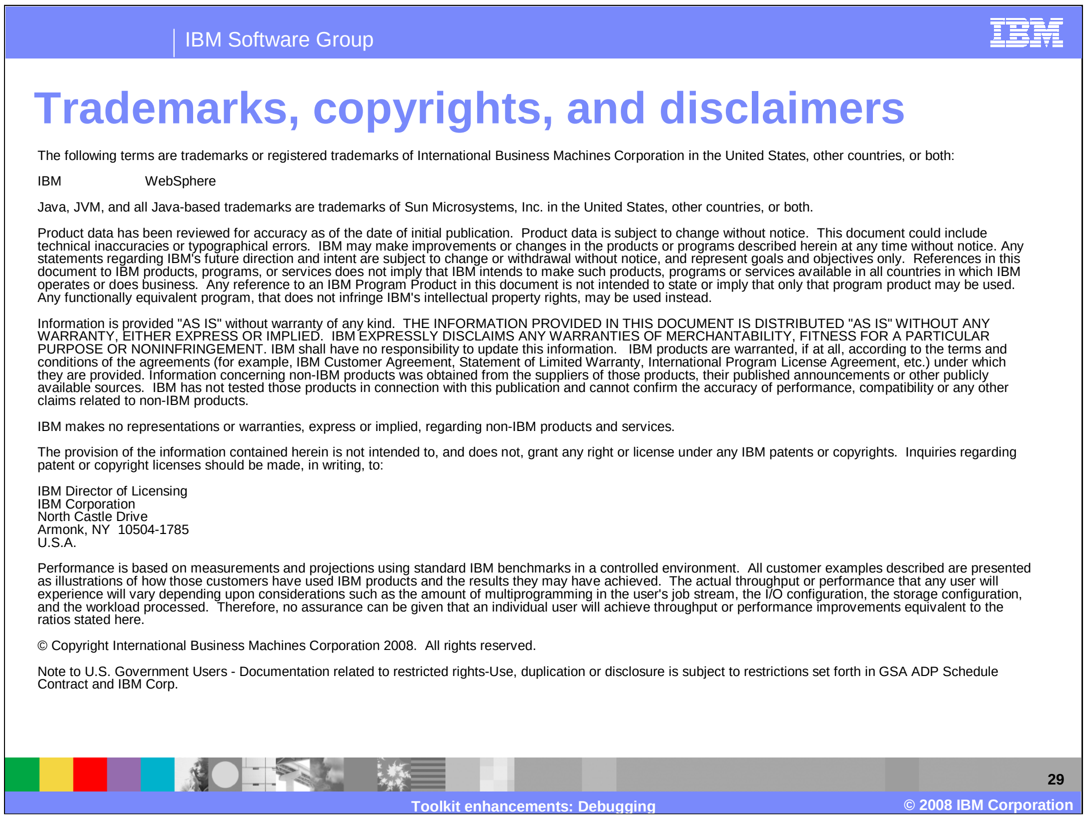
This module is also available in PDF format at: ../WMB61_IEA_Toolkit4_Debug.pdf

28

Toolkit enhancements: Debugging

© 2008 IBM Corporation

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



Trademarks, copyrights, and disclaimers