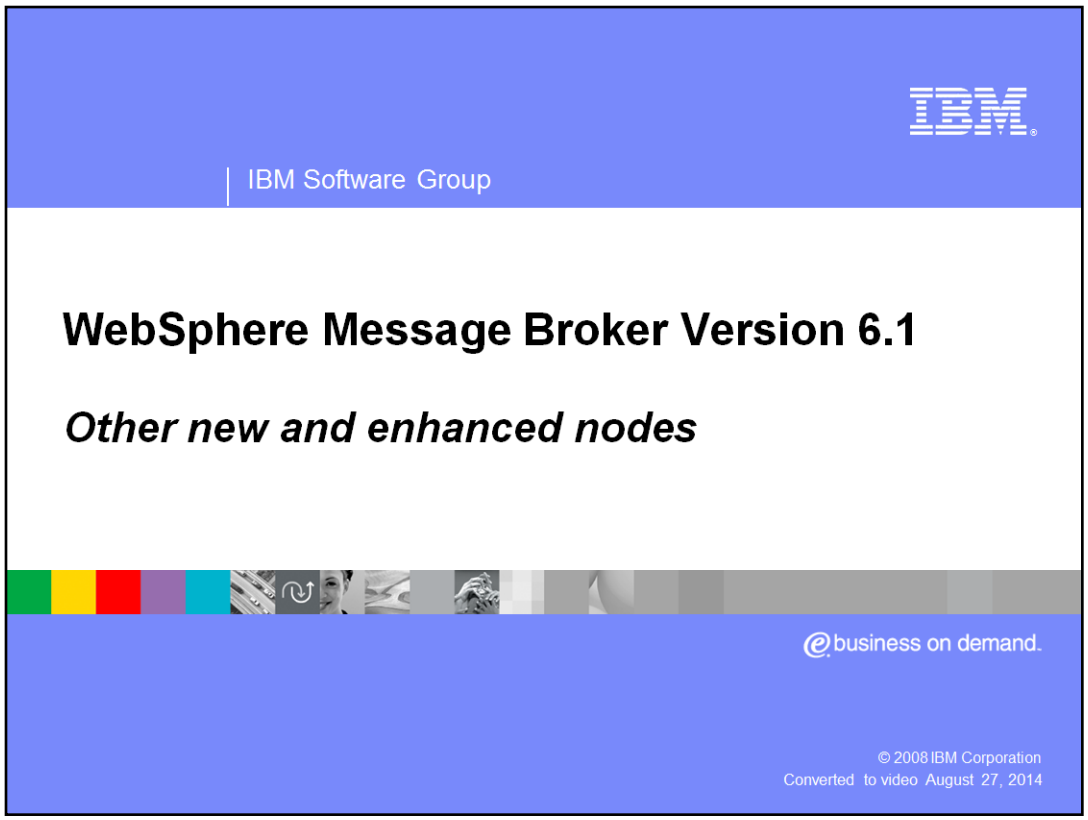
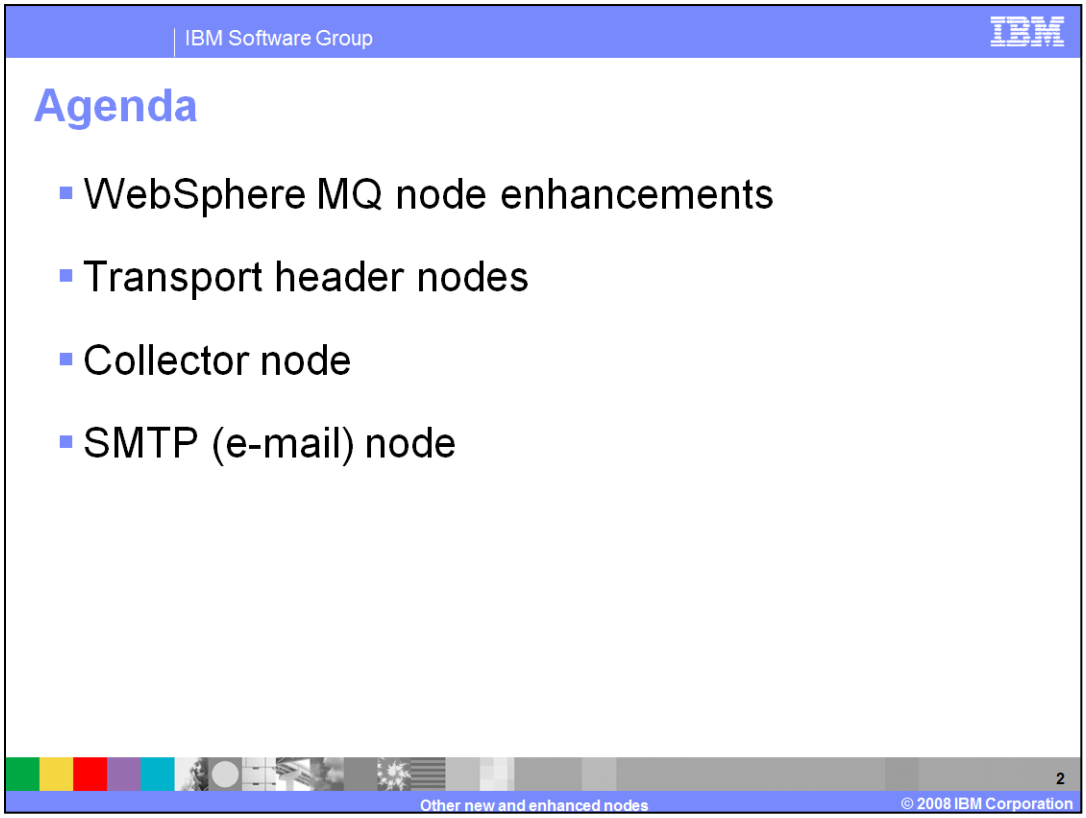


This presentation will give an overall view of the new and enhanced nodes in WebSphere Message Broker Version 6.1.

However, the new SOAP nodes, file nodes, route and database nodes are covered in separate presentations.



This presentation covers the topics shown on this slide.

First, the MQ nodes have been enhanced to implement the Browse capability with WebSphere MQ.

Secondly, a recurring theme with the node enhancements in version 6.1 is the removal of the need for programming as far as possible, either with ESQL or with Java. One of the items that addresses this objective is the provision of new nodes that generate appropriate headers for output operations. This previously required user-written code to achieve this.

Third, a new node called the collector node provides a mechanism to consolidate incoming data from different sources, before passing down the message flow.

And finally, version 6.1 has now introduced full support for the e-mail node.

Browsing MQ messages

- Browse capability added to MQInput and MQGet nodes
- Uses MQ browse (MQGMO)
- Sets *OutputLocalEnvironment.MQ.GET = browsed*
  - or configured MQ parameter location in the case of MQGet node
- Option to reset browse cursor after a period of inactivity
  - This causes messages to be re-browsed
- Threading
  - Each node on a thread will browse in sequence if they are using the same queue.
  - Each thread will browse independently from all other threads.

The MQ Input and MQ Get Nodes have been enhanced to allow a message flow to browse the MQ queue.

The node will browse the first eligible message according to match options specified on the node, or in overrides.

If a message is browsed, the appropriate field in the Local Environment will be set, as shown on this slide. The field will not be written if the message is not browsed. On the MQ Input node, *'Output MQ parameters location'* is not configurable, so it will always write to OutputLocalEnvironment.MQ.GET.

No option is provided to lock the browsed message. If the message flow needs to have exclusive control over a message, it should perform a normal, non-browse read of the queue.

Multiple nodes in the same flow which browse the same queue will advance each others browse cursors. This means, for example, an MQ-Get with a Browse on an MQ-Input node's queue will cause the MQ-Input node to skip a message. To browse on the same queue, both nodes must be using the same message ordering.

As each thread maintains its own browse cursor, messages will not be browsed sequentially across additional flows or flow instances.

Hence, each node on a thread will browse in message sequence, providing they are using the same queue.

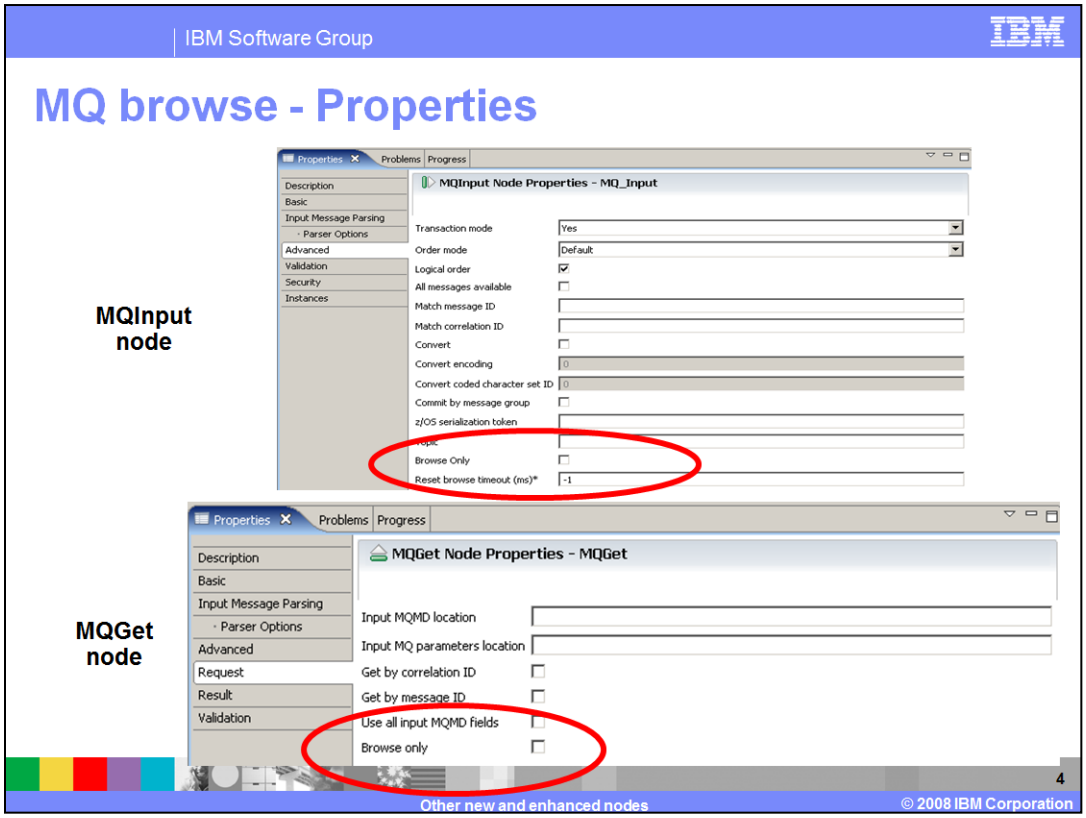
Each thread will browse independently from all other threads.

When an MQ-Input node reaches the end of eligible browse messages, it will wait indefinitely until further messages are put to the queue. Alternatively the '*Reset Browse Timeout*' option can be set to a non-negative interval in milliseconds. Once this period has elapsed after the last message was browsed by the node, the browse cursor will be reset. Any existing eligible messages, including those which may have been skipped, will be browsed or re-browsed.

Skipping can happen if the queue has a Msg-Delivery-Sequence of MQMDS_PRIORITY, *(pronounced M Q M D S Priority).* If a message then arrives on the queue that is of a higher priority than the one currently pointed to by the browse cursor, that message will not be found during the current sweep of the queue using MQGMO_BROWSE_NEXT *(pronounced M Q G M O …)*. It can only be found after the browse cursor has been reset with MQGMO_BROWSE_FIRST, or by reopening the queue. There may also be cases where logical messages are missed if not all segments are available at the time of browse.)

Multiple nodes in the same flow browsing on the same queue will advance each others browse cursors. This means, for example, an MQ-Get browse on an MQ-Input node's queue will cause the MQ-Input node to skip a message. To browse on the same queues, both nodes must be using the same message ordering.

As each thread maintains its own browse cursor, messages will not be browsed sequentially across additional flows or flow instances. If you need more control over a browsed message, perhaps to achieve message locking, you must get and re-put the message.

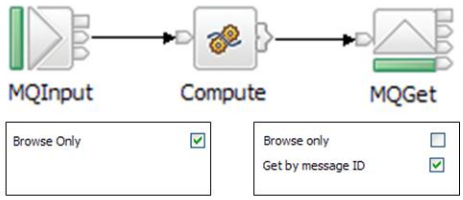


This slide shows the properties of both the MQInput and MQGet nodes.

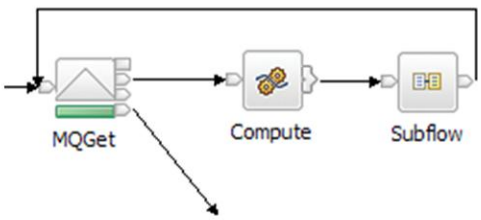
On the MQInput node, the Browse option is on the Advanced tab. On the MQ Input, there is a reset browse timeout, which is specified in milliseconds. If the value is set to “-1”, then the timeout will not be invoked.

On the MQGet node, the Browse option is on the Request tab.

Browse scenarios



- Browse then Get
  - Ensure that the browsed message ID is set in the tree's configured MQMD location



- Browse All
  - Potentially in a single unit of work
  - Be aware of possibility of stack overflows

This slide discusses two scenarios using the MQ Browse option.

The first scenario is a Browse followed by a Get. In this case, the message flow examines the contents of a message before deciding whether to remove it from the queue or not. This can be achieved by an MQInput-MQGet or MQGet-MQGet node combination, with the first node in each case browsing the queue and the second removing it.

To achieve this, set the 'Get by Message ID' option on the second node and ensure the browsed message's message-ID is present in the MQMD. Alternatively, you can use another message tree location as indicated by the request options on the second MQGet node).

In addition, unset the 'Include Message Contents in Output Message Assembly' option on the second node. This option is on the Results tab of the MQ-Get node. This option offers performance improvements, as the message will only be removed from the queue and not completely read and parsed for a second time.

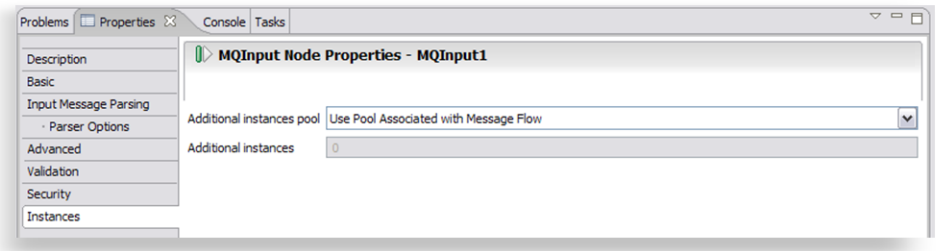
The message flow would require some processing logic between the two nodes to decide if the message should be removed.

Since context information cannot be saved if the message is only browsed, the "Pass_All" option on MQ Output nodes is not valid. The context information will still be written to the message tree and so can be set using "Set_All".

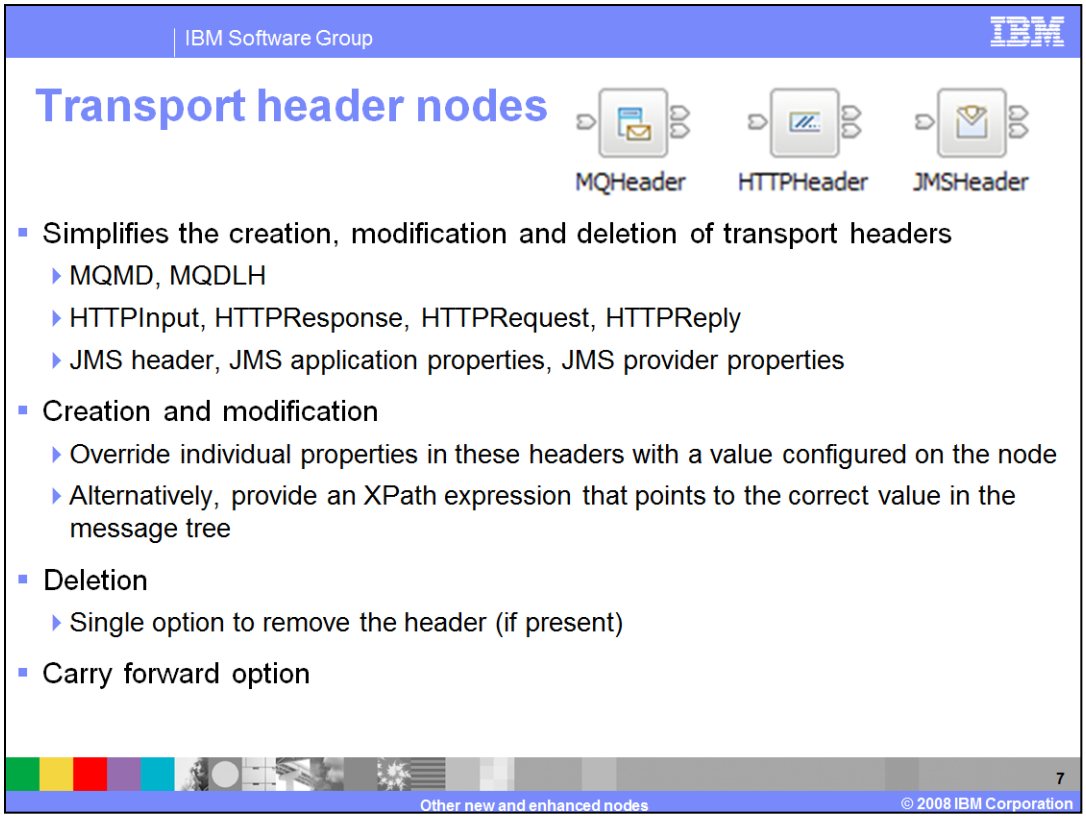
The second scenario enables a message flow to browse many messages, using repeated invocations of the MQ Get node. This should be used with some caution, especially when used within a single transaction, since virtual memory will be used.

MQInput – Additional instances

- Additional instances parameters can now be specified on the MQInput node
- Prevents possible thread starvation when using multiple input nodes in a single flow



A second improvement with the MQInput node is the ability to specify additional instances of the processing thread on the node itself. This can be useful if a message flow monopolizes the available threads.



In versions of Message Broker before version 6.1, in many cases it was necessary to provide additional code to write a message to particular transport. For example, if a flow needed to write a message to an MQ queue, an MQMD header needs to be provided. This has to be done using Java or ESQL code.

Version 6.1 has introduced new transport header nodes, which perform this function without the need to write code.

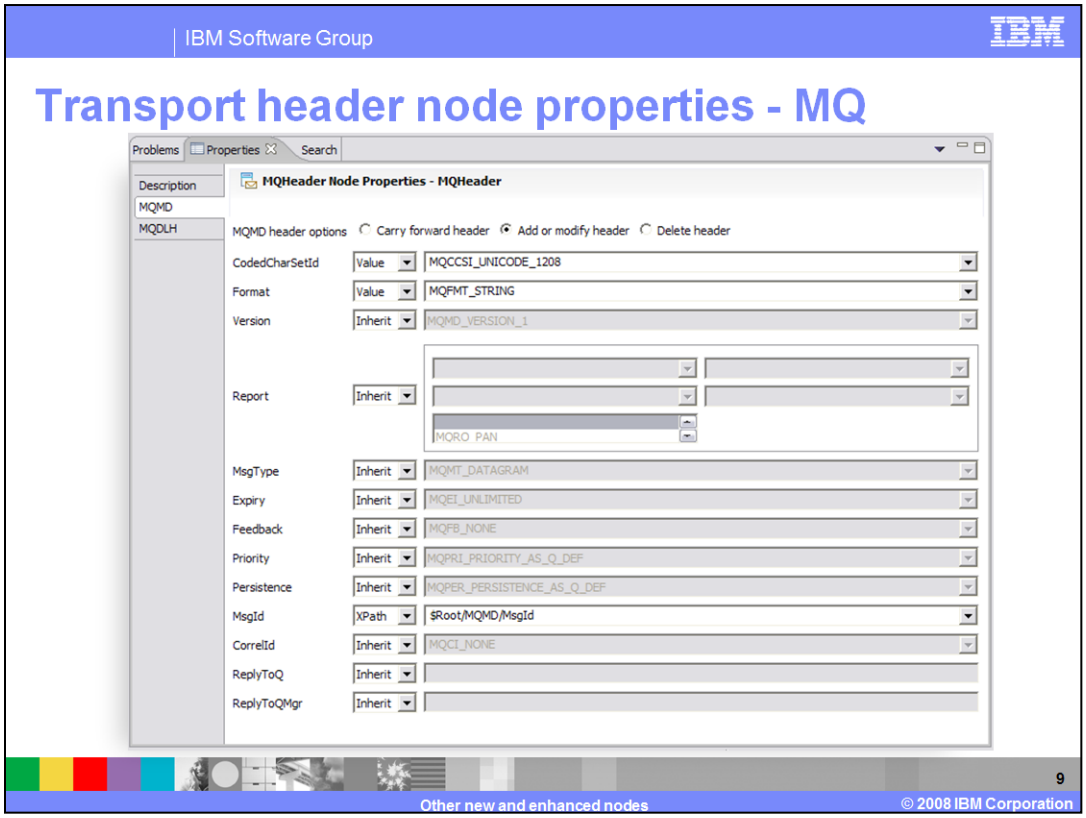
Each node handles all the headers associated with a given transport, so the MQHeader node handles both MQMD *(pronounced M Q M D)* and MQDLH *(M Q D L H)* for example. Where the incoming message has more than one supported header, you can use the 'Carry Forward' option to leave a given header unchanged.

The Transport Header Nodes are currently available as the downloadable SupportPac IA9S.

Section

Transport header nodes

This presentation now discusses the new Transport Header Nodes.

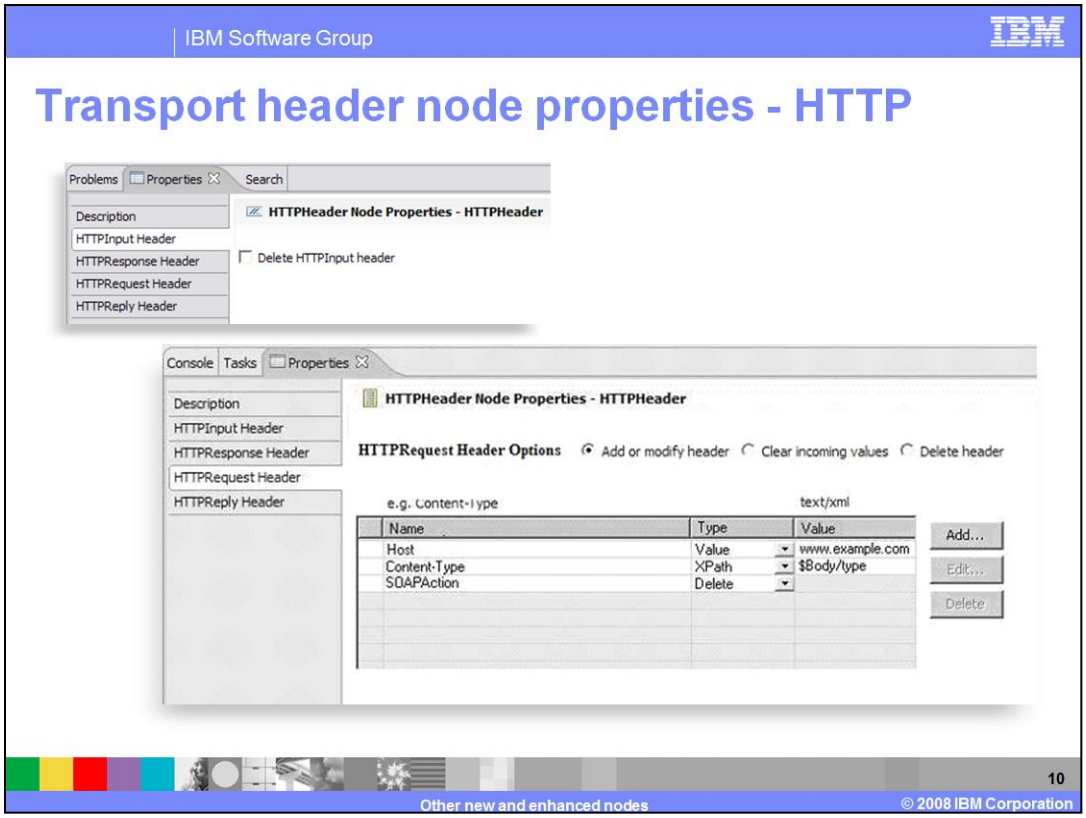


This slide shows the MQMD properties of the MQ transport header node.

In these properties, you can specify a hard-coded value, or you can specify an X-Path expression which can be used to retrieve a value from the message tree.

If you specify the value “inherit”, this means “do not change the value of this property”.

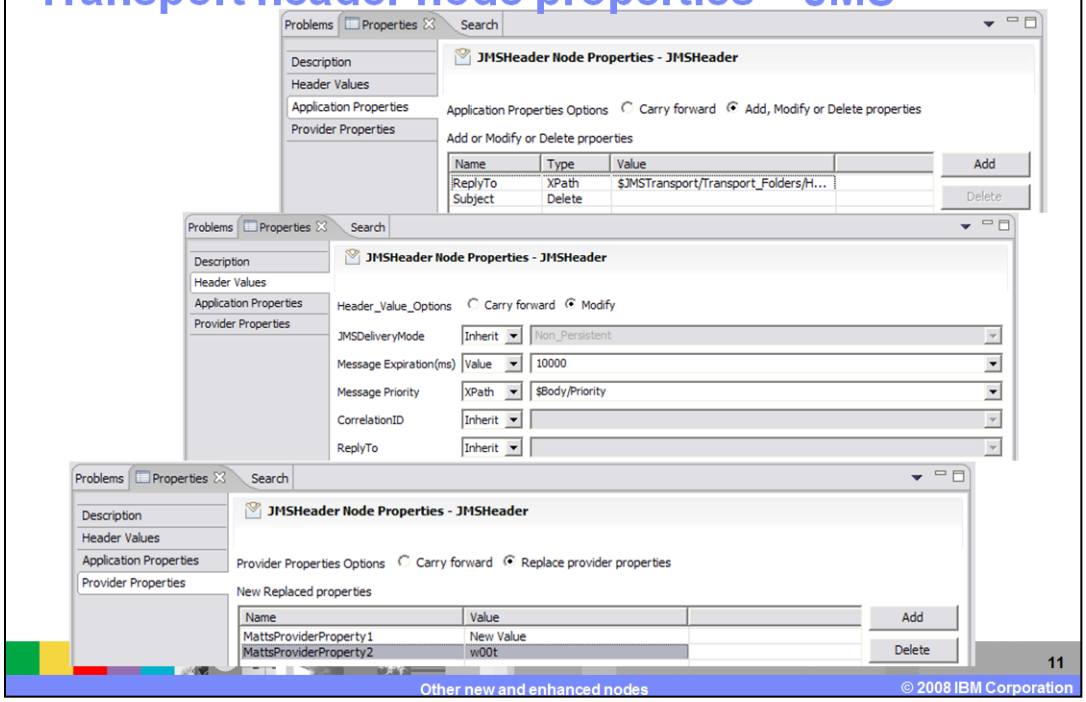
Note at the top of the property window, you can specify whether to add, modify or delete the header.



This slide shows the HTTP properties.

This example shows the headers that are being specified for a HTTP Request. The Host name is set to the hard-coded value of “www.example.com”. The Content-Type is set using an X-Path expression, and the SOAPAction header is being deleted, before control is passed to the HTTP request node itself.

Transport header node properties – JMS

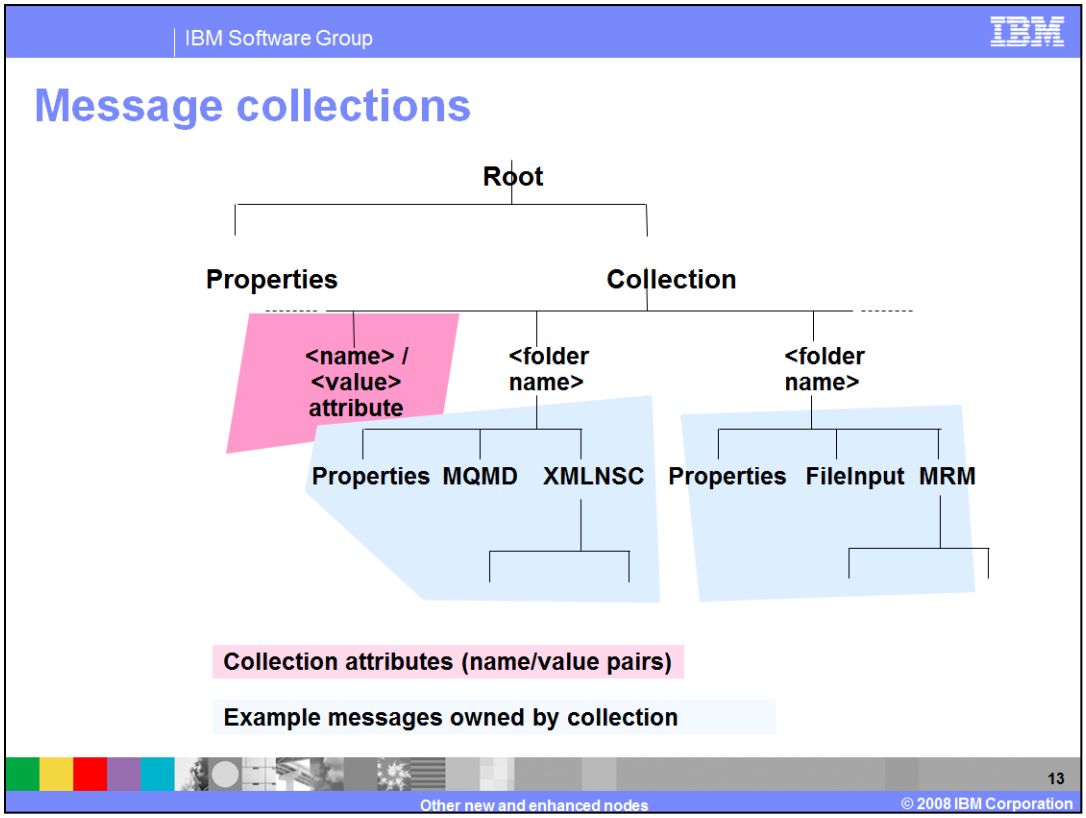


This slide shows the JMS properties.

Section

Collector node

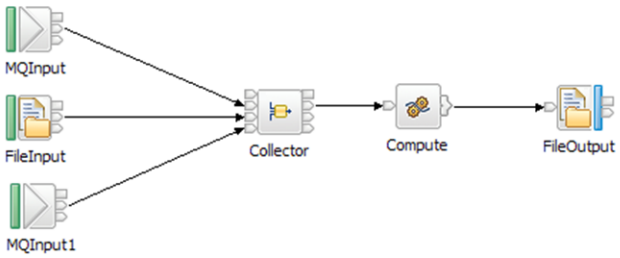
The presentation now moves on to the new collector node.



As preparation to discussing the collector node, Message Broker version 6.1 has introduced the concept of message collections. This is a way to specify multiple messages in the message tree. This technique allows a message flow to address multiple messages within a single message tree. Each message is held in a separate folder under the Collection element of the tree. Each folder element in the collection can contain a different message type, with a structure appropriate to the type of message.

Collector node

- Groups together incoming messages from multiple sources into message collections



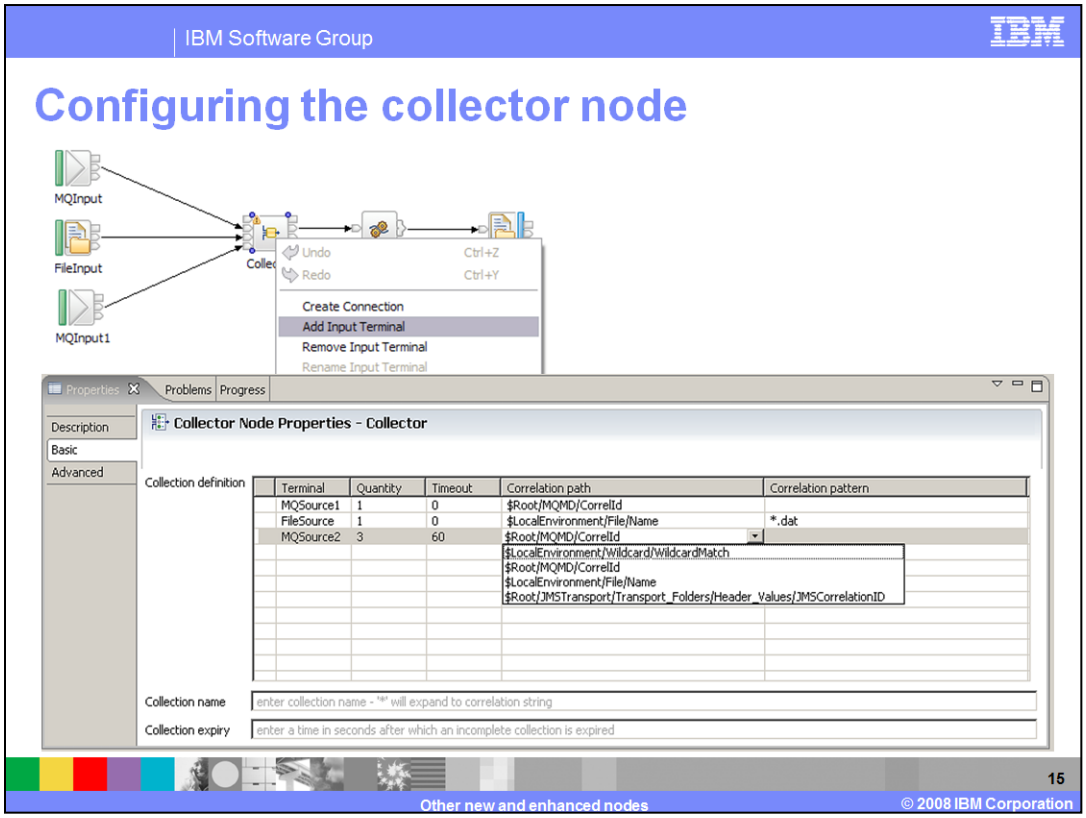
- When a correlated set of messages arrives, the message collection is propagated to the output terminal
- Specify an expiry time for incomplete collections

The collector node is used to collect messages that originate from multiple, different, sources.

This is in contrast to the aggregator node, which combines messages from multiple sources of the same type.

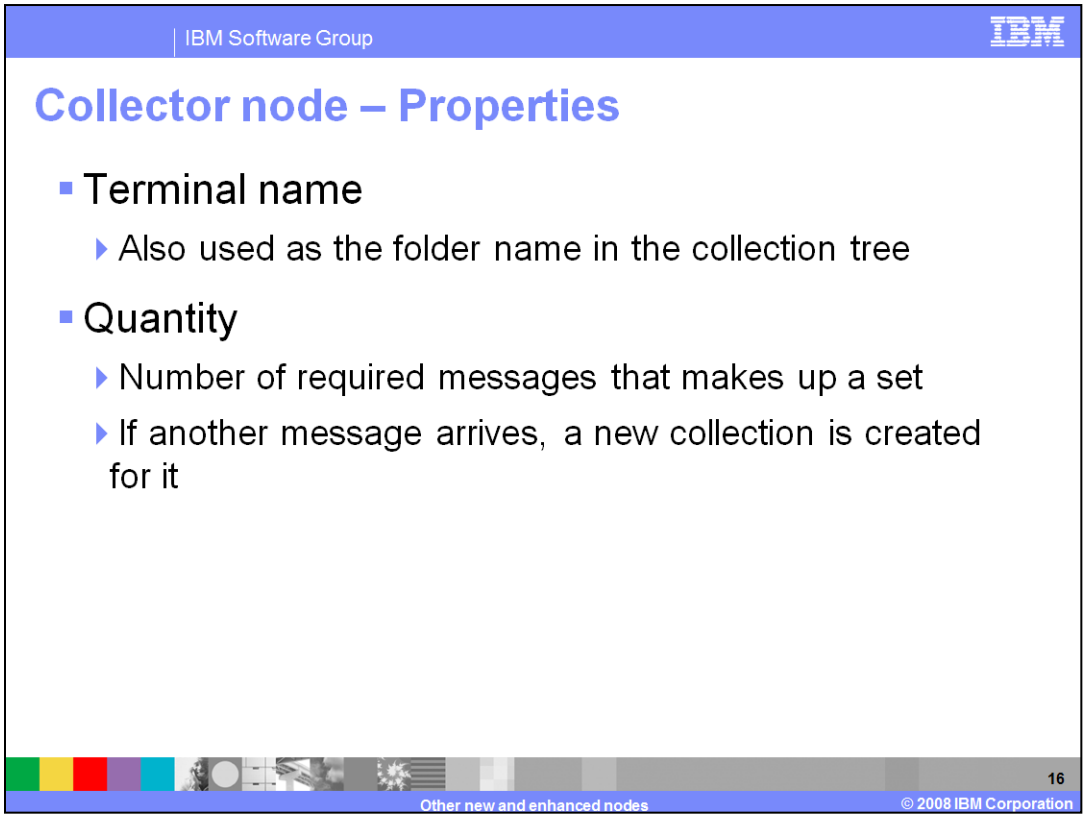
The example on this slide shows the collector node receiving input from two separate MQ queues, combined with input from a file. When the collector node has received messages from all three sources, then the messages will be propagated down to the rest of the message flow, and in this case, will be written out to a file. All of the messages will be available in the message tree, using the collections tree described on the previous slide.

There are several options that can be specified. For example, if two of the input nodes have received messages, but the third one has not, you can specify an expiry time. After this time interval has expired, the collector node will be invoked regardless of whether all inputs have been activated.



The collector node uses dynamic terminals. In this case, these are dynamic input terminals, and are configured by right-clicking the node, and selecting “Add Input Terminal”.

The properties of the collector node are all contained on the Basic tab, which is used to specify the collection definition. The example on this slide has three input terminals, named “MQSource1”, “FileSource”, and “MQSource2”. These are dynamically added to the properties as you add them to the node.



On the collector node, you should define the properties on this slide and the next one.

The terminal name corresponds to the name of the dynamic terminal that was created on the node itself. This is used as the folder name in the collection tree by the message flow that needs access to the data.

The quantity specifies how many messages are required to be received on a particular terminal to make a complete set. This is described as a collection. If an additional message arrives on that particular terminal, a new set, or collection, is created. On the example shown on the previous slide, the first MQ terminal and File terminal are both expecting one message, while the second MQ terminal is expecting three messages. When all three conditions have been met, the collector node will propagate the message tree.

Collector node – Properties (continued)

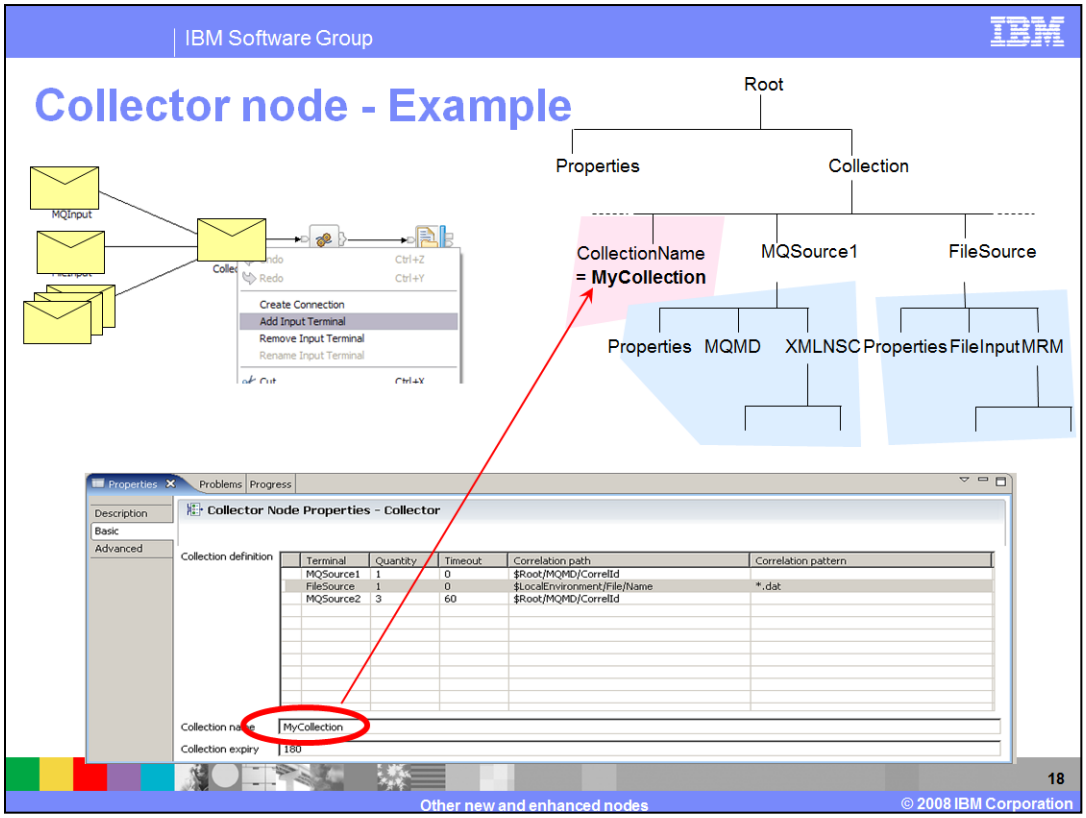
- Timeout
  - When quantity > 1, this is the maximum time (in seconds) to wait for subsequent messages in the set
- Correlation path
  - Location in that message's tree that contains the correlation identifier
- Correlation pattern
  - Substring of the value in the correlation path that contains the correlation identifier
  - *Properties.WildcardMatch* used to store the correlation string

The Timeout property specifies the maximum time to wait, after the first such event has been received. If the timeout value expires before all messages have been received, the message tree will be propagated.

The correlation path is used to specify the path in the incoming message from which to extract a value for the correlation string. Messages are only accepted into a message collection if they have the same correlation string. If the message has a different correlation string, it is offered to the next collection in the queue. If none of the collections accept the message, then a new collection is created with correlation string set to the value of the correlation string in the message.

The correlation pattern is used to specify a pattern to match the contents of a correlation path value against. You must set the Correlation path property before you set the value for Correlation pattern. If you set the correlation pattern, you must use one, and only one, * character, optionally surrounded by other text. For example, "*.dat". If the correlation pattern is blank, the entire text from the correlation path must be matched by the incoming message.

The Properties.WildcardMatch field can be used to store the correlation string. This is one of the pre-configured values in the pull-down, as shown on the previous slide. Alternatively, you can define your own path using an X-Path expression.



This slide shows the same collector node as on the previous slide. In this case, the collection has now been named “MyCollection”. This name is then used in the collection folder, and can be used by the message flow to reference the messages held in that collection.

In addition to specifying a timeout on individual terminals, you can also specify a timeout for the whole collection. This is used to expire the whole collection, in the case where all the expected messages are not received.

Collector node - Control terminal

- Control terminal allows an external resource to trigger the output from the collector node
- 'Event coordination' setting on the node control what to when the control terminal receives an input message
  - Disabled
  - All complete collections
  - First complete collection
- If set to first complete collection, and if a collection is not ready when the control message arrives, it will trigger when the next collection is completed
- The contents of the control message are ignored and discarded

In addition to the dynamic terminals that you have defined, there is a static input terminal called a "control terminal". The control terminal is used to trigger output from the collector node. Any message sent to this terminal will trigger the node to propagate the current message tree to the message flow.

This function makes use of the "Event Coordination" property on the Advanced tab of the collector node properties.

There are three possible values for this property.

Setting the property to "Disabled" means that messages to the control terminal are ignored and collections are propagated when they are complete.

Setting the property to "All Complete Collections" means that complete message collections are held on a WebSphere MQ queue. When a message is received on the control terminal, all the message collections on the WebSphere MQ queue are propagated to the Out terminal.

Setting the property to "First Complete Collection" means that message collections are held on a WebSphere MQ queue. When a message is received on the control terminal, the first message collection on the WebSphere MQ queue is propagated to the Out terminal. If the WebSphere MQ queue is empty when the message is received on the Control terminal, then the next complete message collection is immediately propagated to the Out terminal.

The format of the message sent to the control terminal is not examined, and the message itself is discarded after use.

Collector node - Managing the state queues

- All in-flight collections are held in state queues (two per broker)
  - SYSTEM.BROKER.EDA.COLLECTIONS
  - SYSTEM.BROKER.EDA.EVENTS
- If an expiry interval is not specified OR if using event coordination, the state queues can grow indefinitely.
  - This could eventually cause one or both of the queues to exceed its maximum queue depth.
  - In this case a BIP4704 exception will be thrown.
  - Manage this situation by either increasing the queue depths or by setting a realistic expiry for the collections.

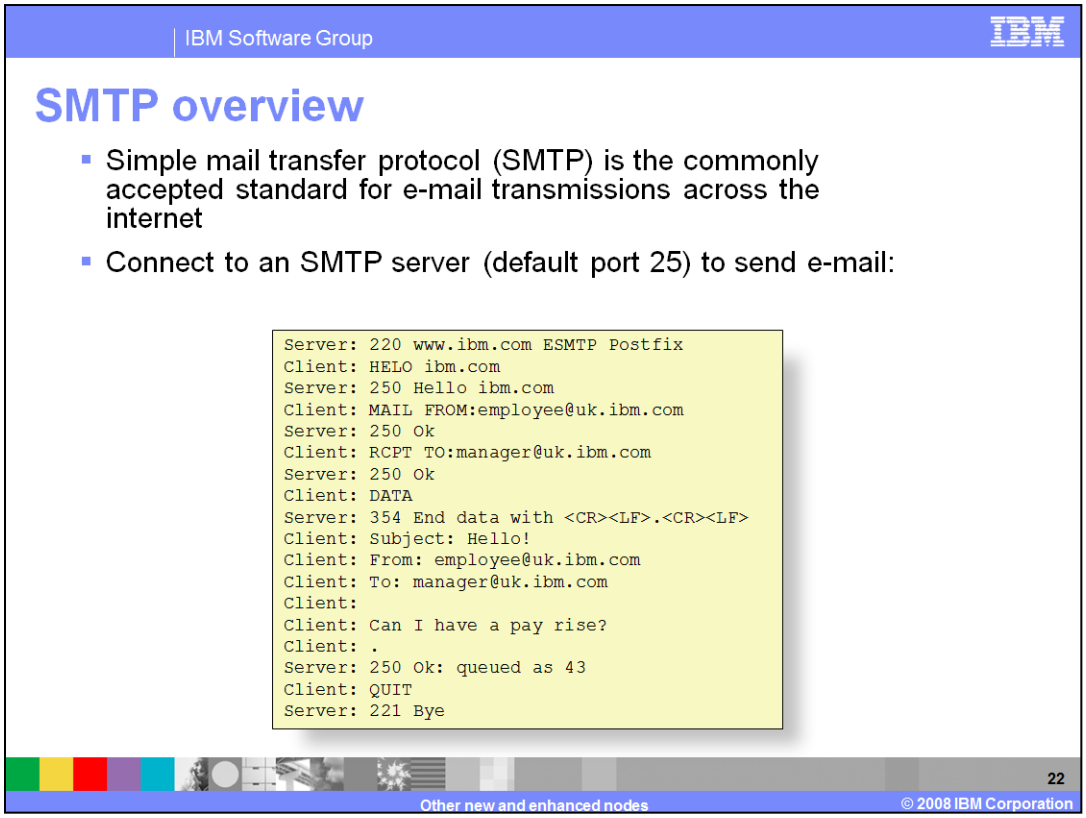
The collector node stores messages on WebSphere MQ queues. Two queues per broker are used.

In certain circumstances, these queues could grow indefinitely. If you use the collector node in such a way that may allow this, you should monitor the usage and size of these queues, and make appropriate adjustments if needed.

Section

SMTP (e-mail) node

The final topic in this presentation is the e-mail node.



To send e-mail, SMTP is a commonly accepted standard, and is used by an e-mail client to connect to an e-mail server and perform the e-mail operation.

The dialogue shown on this slide shows the interaction between an e-mail client and server system.

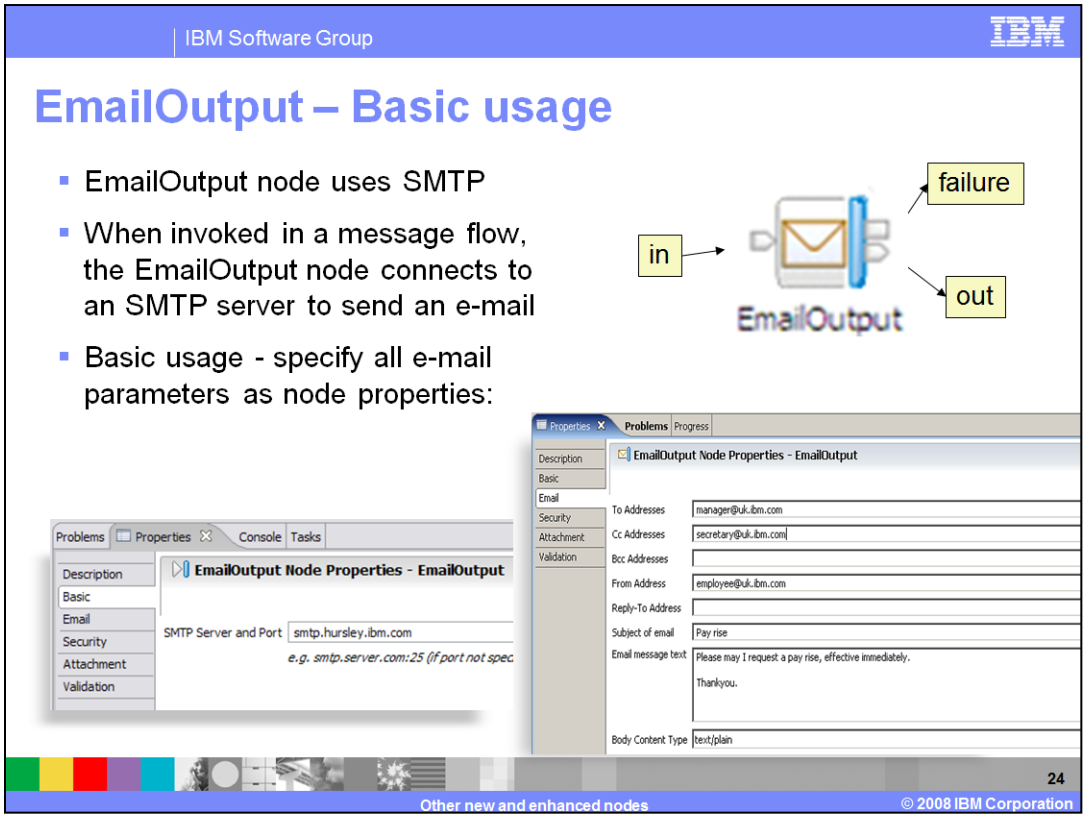
MIME (Multipurpose internet mail extensions)

- MIME is a specification for enhancing the capabilities of standard Internet electronic mail.
  - Typically used for encoding binary data
  - Designed for e-mail, but increasingly popular for general messaging
  - It offers a simple standardized way to represent and encode a wide variety of media types for transmission over the Internet
  - Messages can consist of one or more 'parts', each of a defined 'content type'
  - A message with more than one part is a 'Multipart MIME' message
- RFC1521 and RFC1522 (1992) from http://www.w3.org

```
Content-type: multipart/mixed; boundary="delimiter"
MIME-version: 1.0
--delimiter
Content-type: text/plain
This is the body of the message.
--delimiter
Content-type: application/octet-stream
Content-transfer-encoding: base64
SFIO4viuhewfv9823rt092u3ufj09WQd3539247rtutpivewnb09w4
sB&uwqFTFG438UW0J4Gz394JG203BNT9047T09W34UG0A93W4JAGO3
--delimiter--
```


SMTP is often extended with the "MIME" options. "MIME" is a extension to the e-mail standards which allows binary data to be transmitted. It is typically used with e-mail messages that are transmitted using the internet.

SMTP and MIME are the two key e-mail components that are used with the e-mail output node.

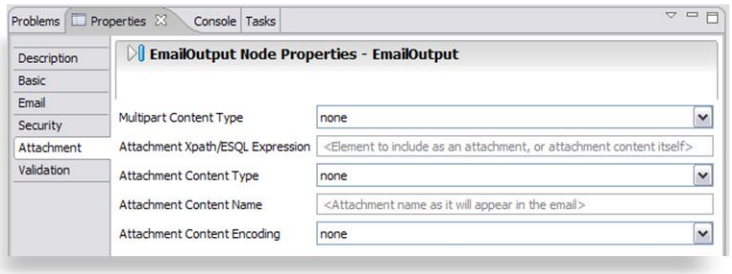


The EmailOutput node is illustrated on this slide.

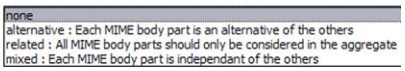
On the basic properties, you specify the name of the SMTP server to which the broker will attach. If not specified, the port is assumed to be 25, which is the normal default for SMTP servers.

On the e-mail, you can specify details of the actual e-mail that you are going to send. This includes the recipient name, addresses of recipients who should received a copy, and other common e-mail information. These values can be specified as variables, described on the next slides.

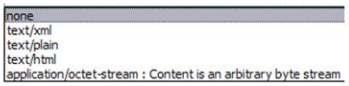
Attachments



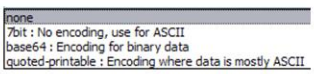
- Multipart content type



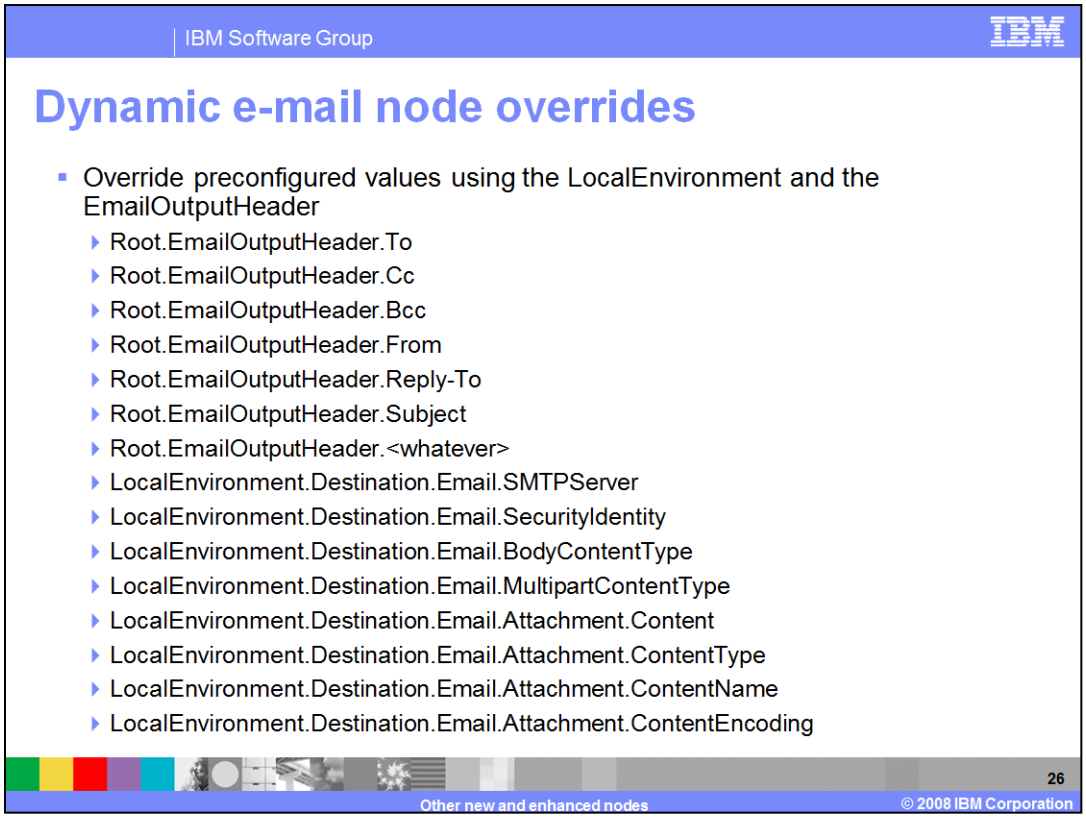
- Attachment content type



- Attachment content name



The attachment tab allows you to specify any attachment you are going to include with the e-mail. This is done using the MIME information discussed earlier.



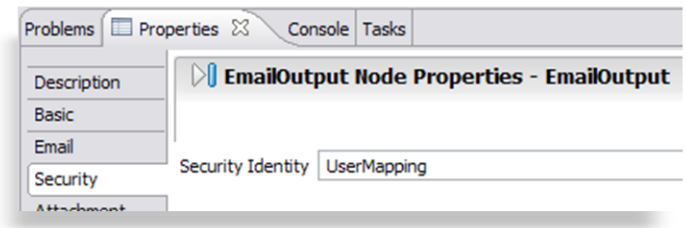
The values of the properties set in the e-mail tab can be over-ridden with values specified in the “EmailOutputHeader” part of the Root tree. These values are shown on this slide, and will be used in preference to any hard-code values on the node. For the fields which specify the e-mail addresses, these should be specified as a comma-separated list.

You can add your own headers using “Root.Email.Output.Header.whatever” field. This can be used to create a header of any value, to suit your own requirements.

Similarly, the e-mail properties in the local environment can be used to over-ride the properties specified on the Basic tab in the e-mail node.

Logging on to SMTP servers

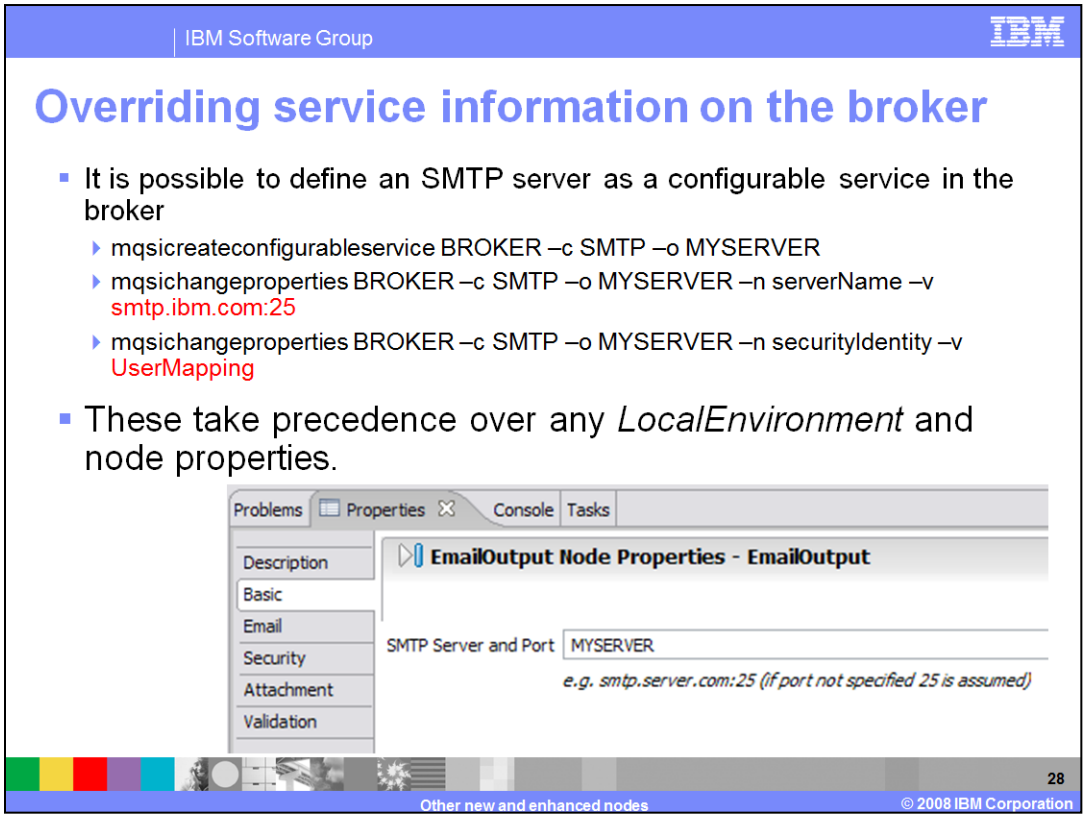
- Some SMTP servers require a username and password
- Define on the runtime broker
  - mqsisetdbparms BROKER –n smtp::UserMapping –u USER –p PASS
- Reference the mapping in the Security tab, overridden using LocalEnvironment.Destination.Email.SecurityIdentity



In some cases, the SMTP server requires a username and password to connect.

This information is specified using the “mqsi set db parms” command. This command specifies the name of the Security Identification, which is then referenced on the security tab of the e-mail node. In the example shown on this slide, this is set to “UserMapping”.

As well as referring directly to this on the e-mail node properties, this can also be over-ridden by setting the “Security identity” property in the Local Environment.



As with many external services, the SMTP server can be specified as a "Configurable Service" within the broker. In the example shown on this slide, this service has been named "MY-SERVER".

This definition can then be referenced in the e-mail properties, on the Basic tab, by referring to this Configurable Service.

In this example, the configurable service has been specified to refer to an SMTP server called "smtp.ibm.com", using port 25. Additionally, the security setting has been changed to use the security definition called "User mapping".

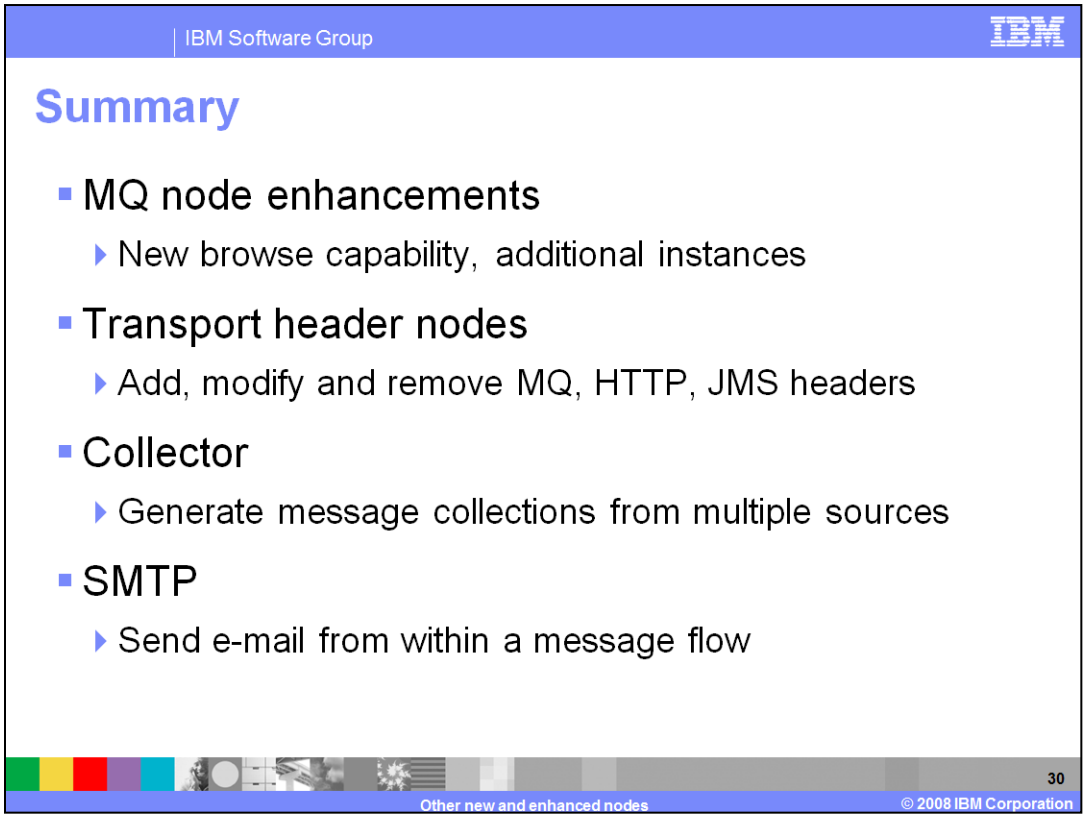
If you specify any properties in this way, these will take precedence over any properties specified in the Local Environment, or in the node itself.

The main reason to specify these properties in this way is to allow these properties to be set at runtime. This then avoids the need to change properties on a node or message flow, and avoids the need to deploy a message flow.

Direct e-mail of MIME messages

- Instead of using the Attachments panel, you can pass a MIME message directly to the EmailOutput node
- The MIME parser parses the message and the content is e-mailed directly to the recipients in the EmailOutputHeader

Finally, instead of specifying e-mails using the MIME properties in the Attachments tab, you can send MIME messages direct to the EmailOutput node. If you use the MIME parser to create the message, the e-mail will already have the expected format for e-mail messages with a MIME attachment.



In summary, this presentation has covered the enhancements made to MQ in version 6.1, and has introduced the new transport header, collector and e-mail Nodes.

Remember, this presentation has not covered all of the new nodes in version 6.1. Separate presentations cover the routing, file, adapters and Web services nodes.

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.com 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.