IBM Software Group

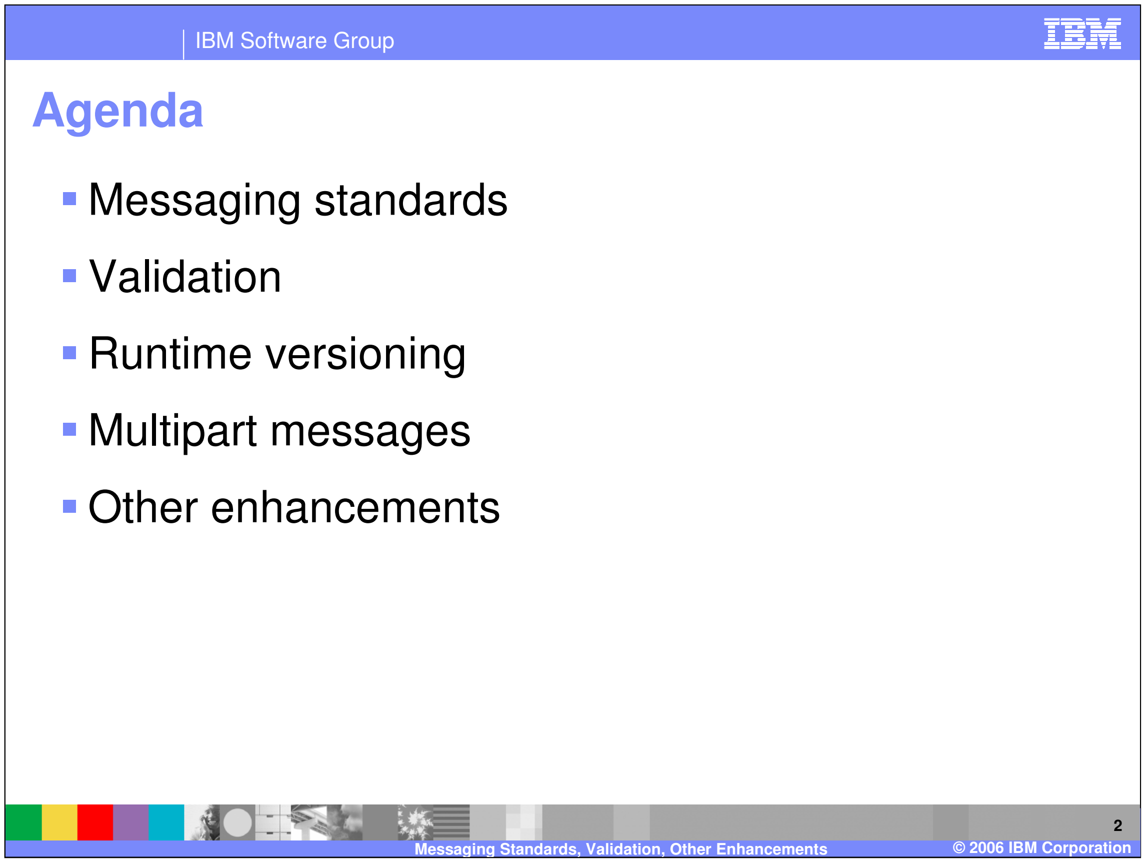
WebSphere® Message Broker V6

Messaging Standards, Validation, Other Enhancements

@business on demand.

Updated October 6, 2006

This presentation discusses changes in Messaging Standards, Validation and Other Messaging Enhancements implemented in WebSphere Message Broker V6.



Agenda

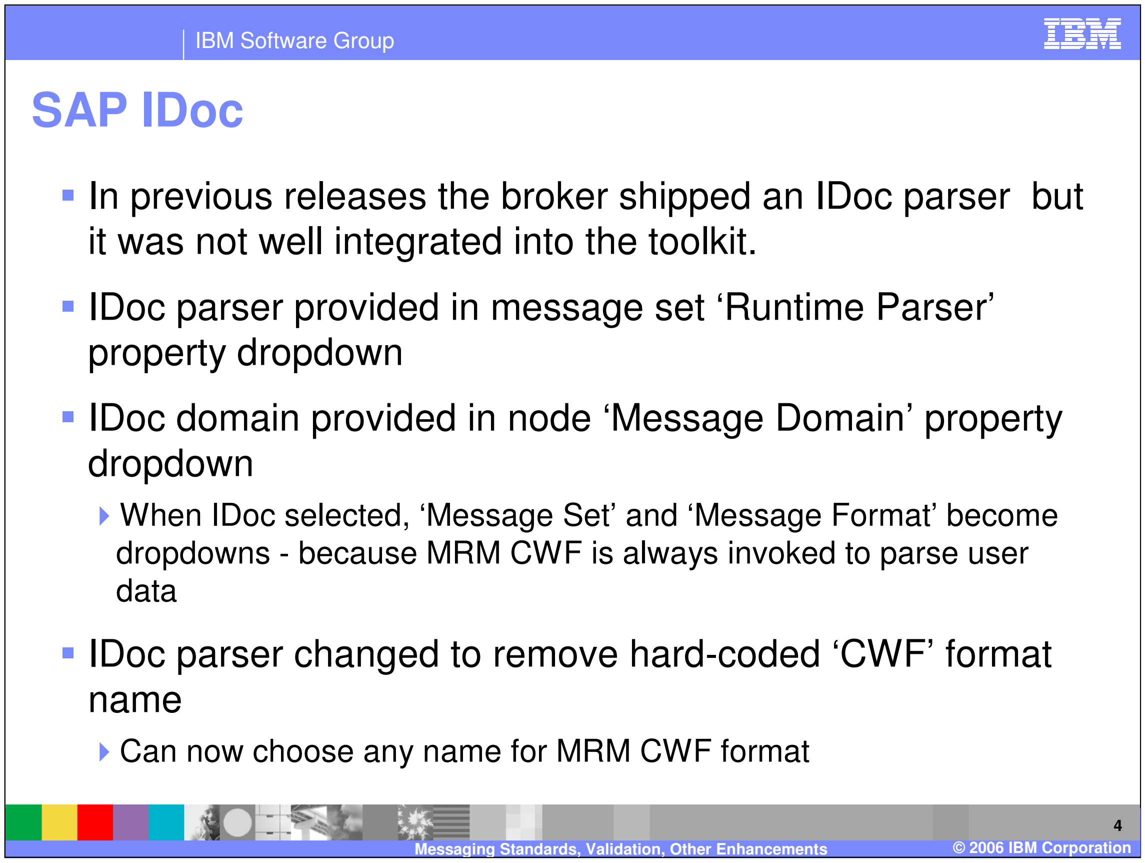
- Messaging standards
- Validation
- Runtime versioning
- Multipart messages
- Other enhancements

The topics presented in this presentation include Messaging Standards, Validation, Runtime Versioning, Multipart Messages, and Other Enhancements.

Section

Messaging standards

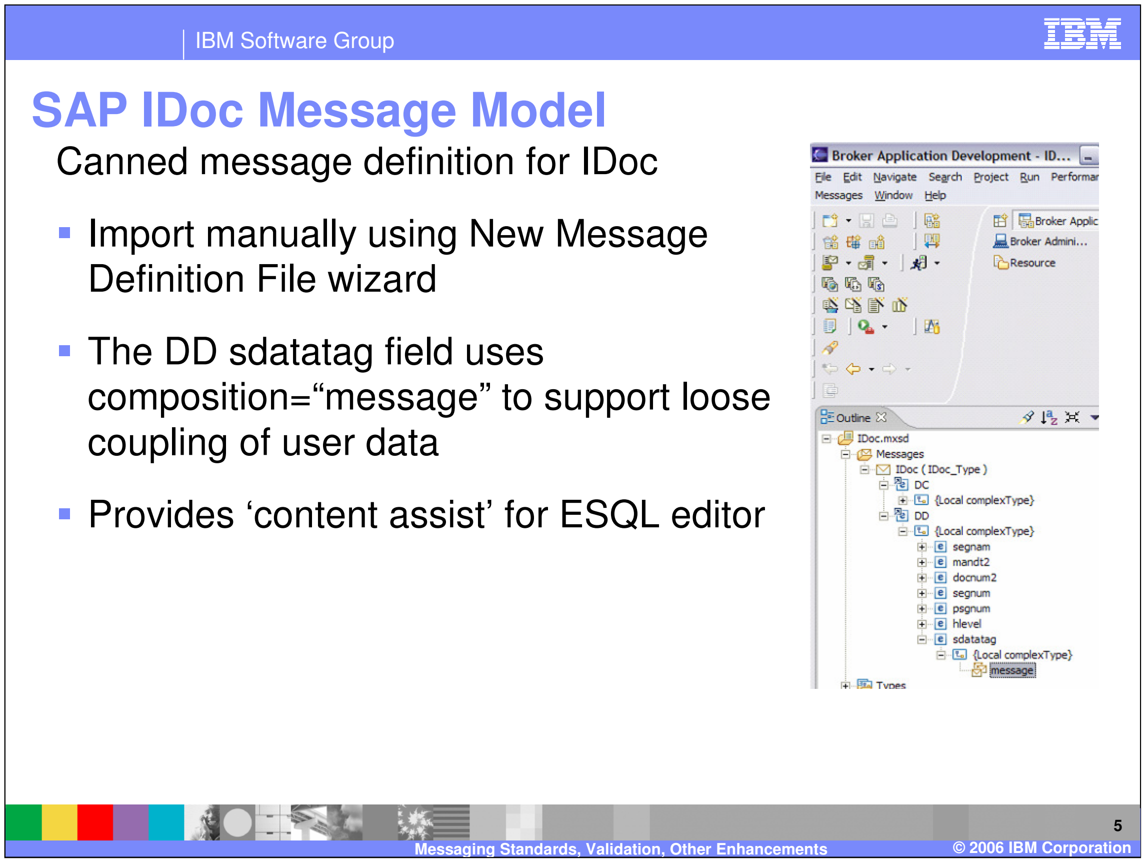
This section discusses Messaging Standards.



SAP IDoc

- In previous releases the broker shipped an IDoc parser but it was not well integrated into the toolkit.
- IDoc parser provided in message set ‘Runtime Parser’ property dropdown
- IDoc domain provided in node ‘Message Domain’ property dropdown
  - When IDoc selected, ‘Message Set’ and ‘Message Format’ become dropdowns - because MRM CWF is always invoked to parse user data
- IDoc parser changed to remove hard-coded ‘CWF’ format name
  - Can now choose any name for MRM CWF format

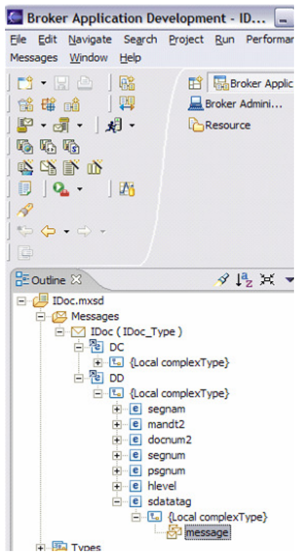
In WebSphere Message Broker V5, the broker was shipped with an SAP IDoc parser, but it was not well integrated into the Message Broker Toolkit. The SAP IDoc parser has been enhanced in V6. The SAP IDoc parser is provided in the message set “Runtime Parser” property dropdown. The IDoc domain is provided in the node’s “Message Domain” property dropdown. Note that when IDoc is selected, “Message Set” and “Message Format” are also selectable dropdowns; this is because the Message Repository Manager’s Custom Wire Format is always invoked to parse user data. The IDoc parser has been changed in V6 to remove the hard-coded “CWF” format name. You can now choose any name for your Custom Wire Format name.



SAP IDoc Message Model

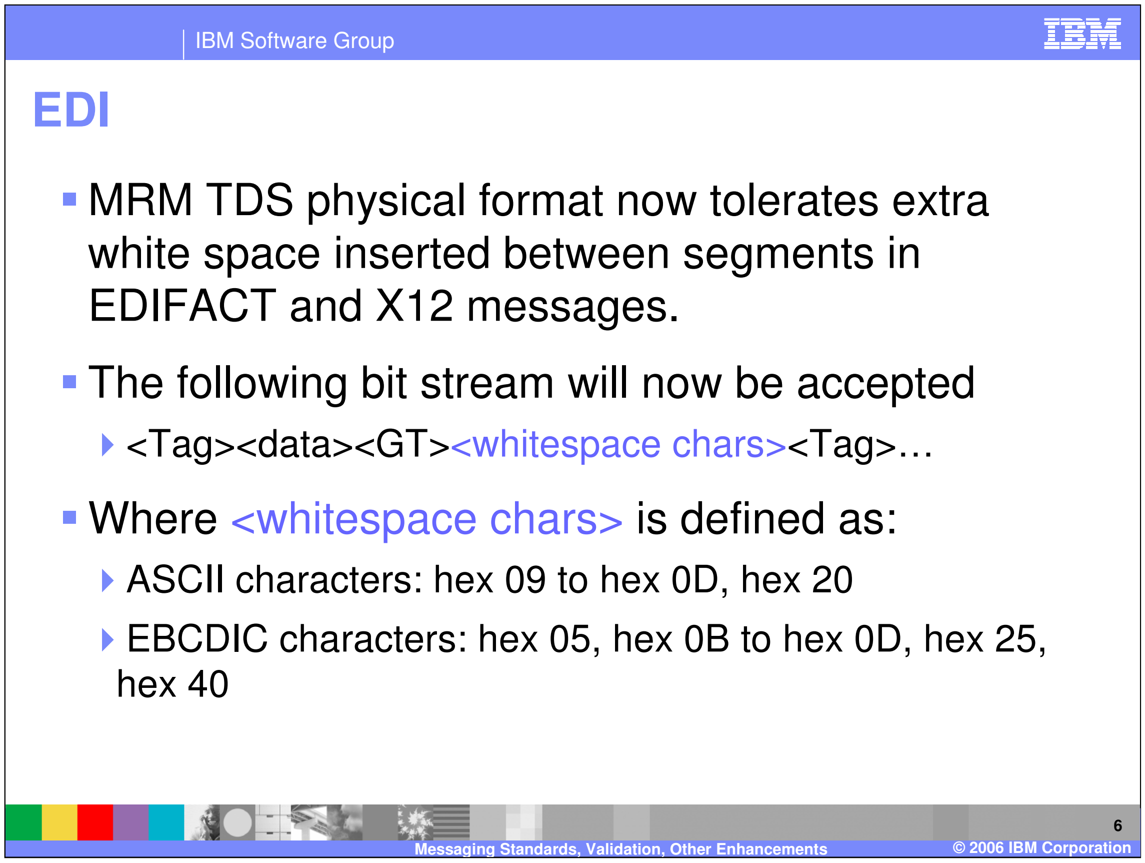
Canned message definition for IDoc

- Import manually using New Message Definition File wizard
- The DD sdatatag field uses composition="message" to support loose coupling of user data
- Provides 'content assist' for ESQL editor



In V6, a SAP IDoc Message Model is provided. This provides canned message definitions for IDoc configurations. You can import configuration information for the SAP IDoc Message Model using the New Message Definition File wizard. One feature of the IDoc Message Model is the DD sdatatag field, which uses composition = "message" to support the loose coupling of user data.

Within the Message Model, content assist is available within the ESQL editor.



EDI

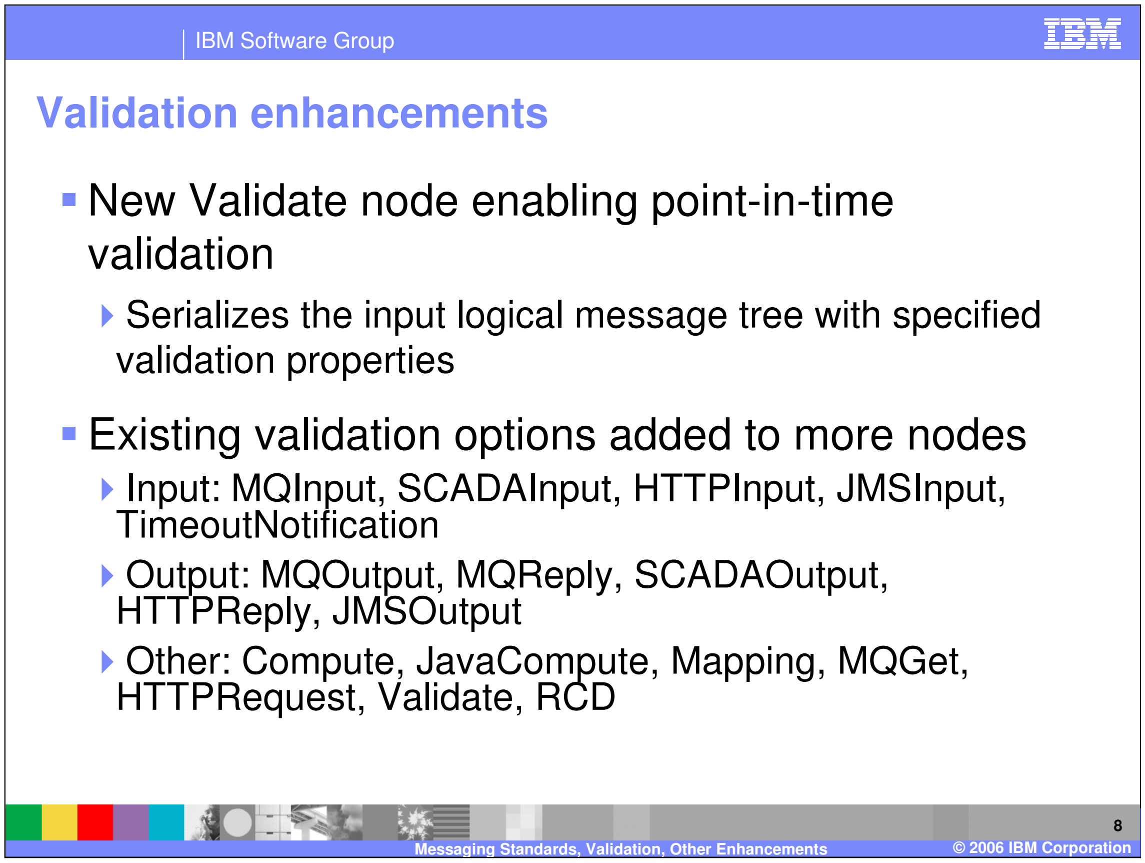
- MRM TDS physical format now tolerates extra white space inserted between segments in EDIFACT and X12 messages.
- The following bit stream will now be accepted
  - <Tag><data><GT><whitespace chars><Tag>…
- Where <whitespace chars> is defined as:
  - ASCII characters: hex 09 to hex 0D, hex 20
  - EBCDIC characters: hex 05, hex 0B to hex 0D, hex 25, hex 40

In WebSphere Message Broker V6, the Message Repository Manager Tag Delimited String format (TDS) has been enhanced. The TDS format now tolerates extra whitespace inserted between segments in EDIFACT and X12 messages, commonly done to improve readability. These allowed whitespace characters include the ASCII characters hex 09 to hex 0D, and hex 20. For EBCDIC, the allowable values are hex 05, 0B, 0D, 25, and 40.

Section

Validation enhancements

This section discusses the validation enhancements made to WebSphere Message Broker V6.



Validation enhancements

- New Validate node enabling point-in-time validation
  - Serializes the input logical message tree with specified validation properties
- Existing validation options added to more nodes
  - Input: MQInput, SCADAInput, HTTPInput, JMSInput, TimeoutNotification
  - Output: MQOutput, MQReply, SCADAOutput, HTTPReply, JMSOutput
  - Other: Compute, JavaCompute, Mapping, MQGet, HTTPRequest, Validate, RCD

A new Validate node has been introduced in V6 that enables point-in-time validation. The point-in-time validation options are:

1) On Demand - validation of a field in the message is delayed until it is parsed by partial parsing.

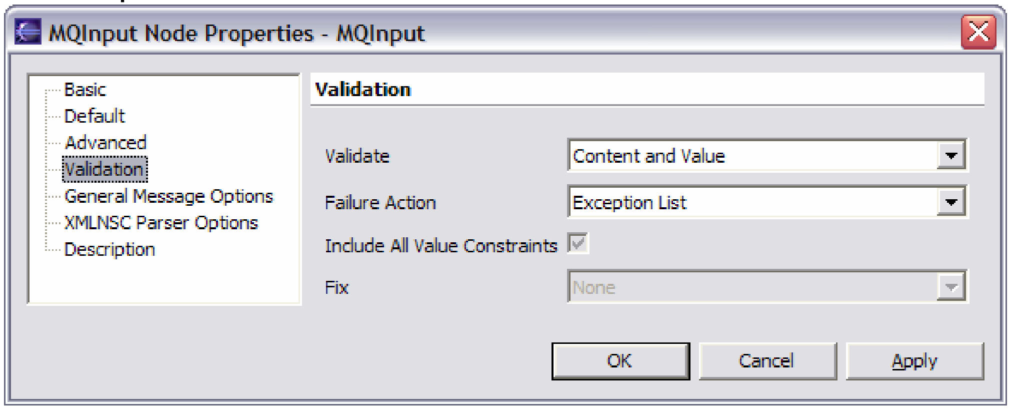
2) Immediate - partial parsing is overridden, and everything in the message is parsed and validated except those complex types with a Composition of Choice or Message that can not be resolved at the time

3) Complete - partial parsing is overridden, and everything is parsed and validated. Complex types with a Composition of Choice or Message that can not be resolved at the time cause a validation failure.

In V6, validation options have been added to Input nodes: MQInput, SCADAInput, HTTPInput, JMSInput, TimeoutNotification. Also, validation options have been added to Output nodes: MQOutput, MQReply, SCADAOutput, HTTPReply, and JMSOutput. Finally, other nodes to which validation options have been added are: Compute, JavaCompute, Mapping, MQGet, HTTPRequest, Validate, and ResetContentDescriptor (RCD).

Validation enhancements – Exception list

- New ‘Exception List’ option for ‘Failure Action’ property
  - Throws an exception if validation failures encountered
  - But only when the current parsing or writing operation has completed



MQInput Node Properties - MQInput

- Basic
- Default
- Advanced
- Validation
- General Message Options
- XMLNSC Parser Options
- Description

Validation

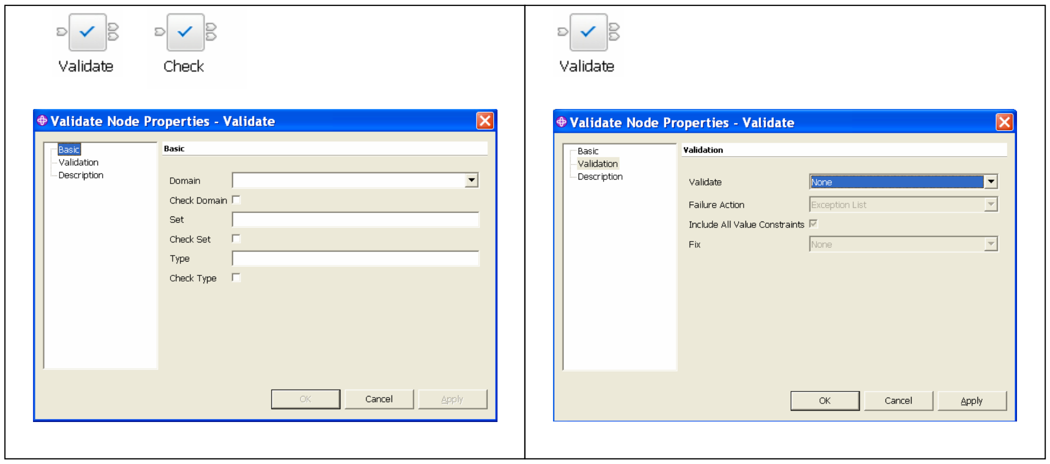
Validate	Content and Value
Failure Action	Exception List
Include All Value Constraints	✓
Fix	None

In WebSphere Message Broker V6, a new Failure Action of “Exception List” option has been added for validation. The “Exception List” action will throw an exception after the current parsing or writing operation has completed if validation failures occur.

Use the “Exception List” setting if you want processing of the message to halt if a validation failure occurs, but you want to see the full list of failures encountered.

Validation enhancements – Graphical view

- New Validate node same as Check node except “Validation” properties
- Check node retained for migration purposes



Basic

Domain
Check Domain ☐
Set
Check Set ☐
Type
Check Type ☐

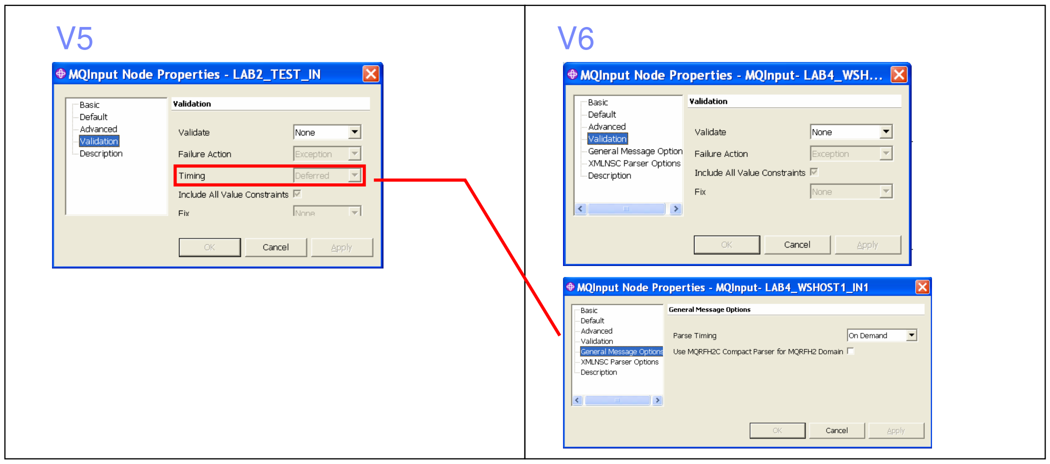
Validation

Validate: None
Failure Action: Exception List
Include All Value Constraints ☑
Fix: None

The chart shows the “Basic” and “Validate” settings for the new Validate node. The validate node supersedes the functionality of the check node and additionally includes validation properties, which were not available with the check node. For migration compatibility reasons, the check node has been retained in V6.

Validation enhancements – Other nodes

- Validation timing for other types of nodes enhanced
- Point-in-time properties settings are found in General Message Options in V6



V5 – Validation

Validate	None
Failure Action	Exception
Timing	Deferred
Include All Value Constraints	✓
Fix	None

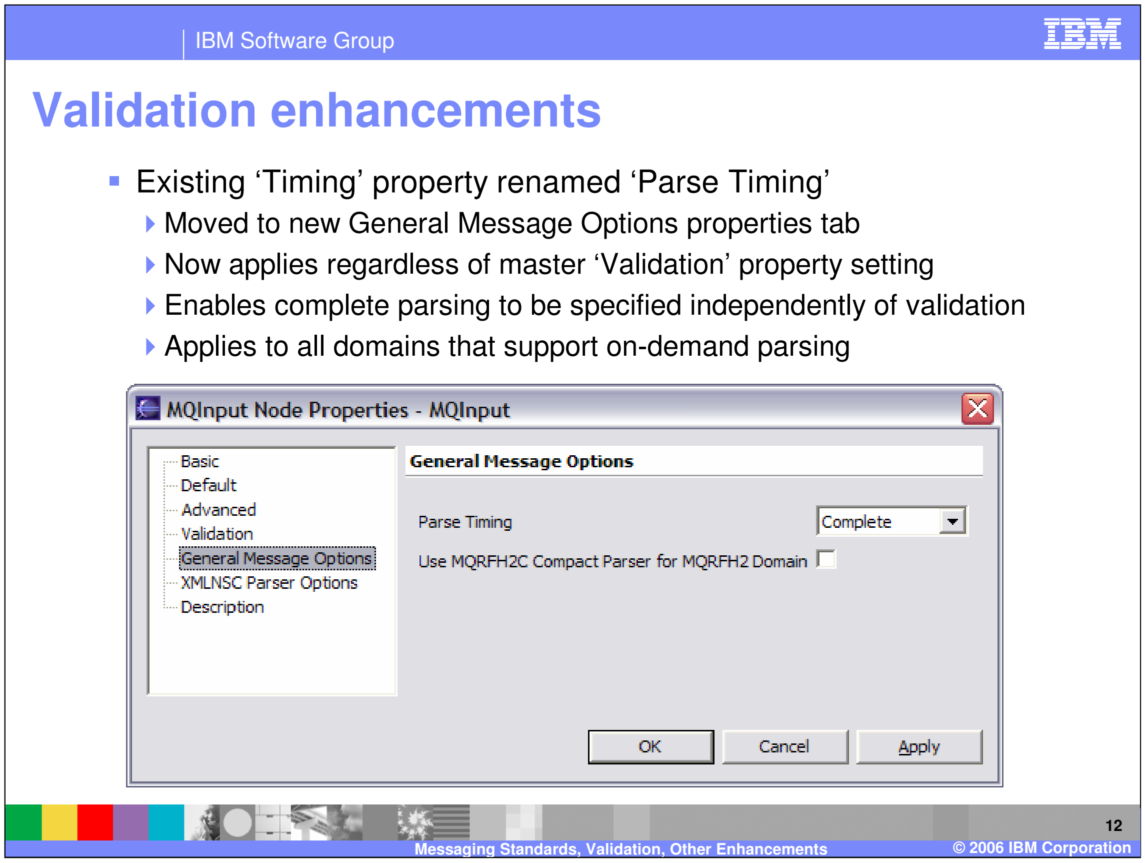
V6 – Validation

Validate	None
Failure Action	Exception
Include All Value Constraints	✓
Fix	None

V6 – General Message Options

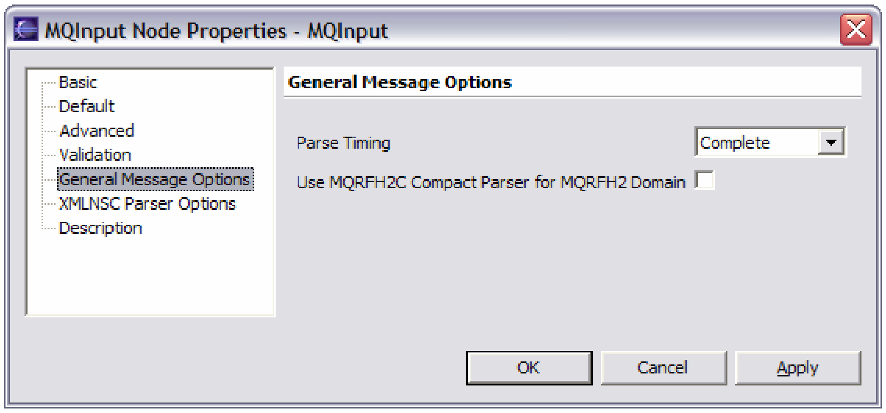
Parse Timing	On Demand
Use MQRFH2C Compact Parser for MQRFH2 Domain	☐

Other types of nodes that now have validation function also allow the new “Exception List” Failure Action. In the example above, you see the V5 MQInput node with the property settings available for validation. The “Timing” property in the V5 MQInput node is now called “Parse Timing” in V6 and is provided under “General Message Options”.



Validation enhancements

- Existing ‘Timing’ property renamed ‘Parse Timing’
  - Moved to new General Message Options properties tab
  - Now applies regardless of master ‘Validation’ property setting
  - Enables complete parsing to be specified independently of validation
  - Applies to all domains that support on-demand parsing



MQInput Node Properties - MQInput

General Message Options

Parse Timing: Complete

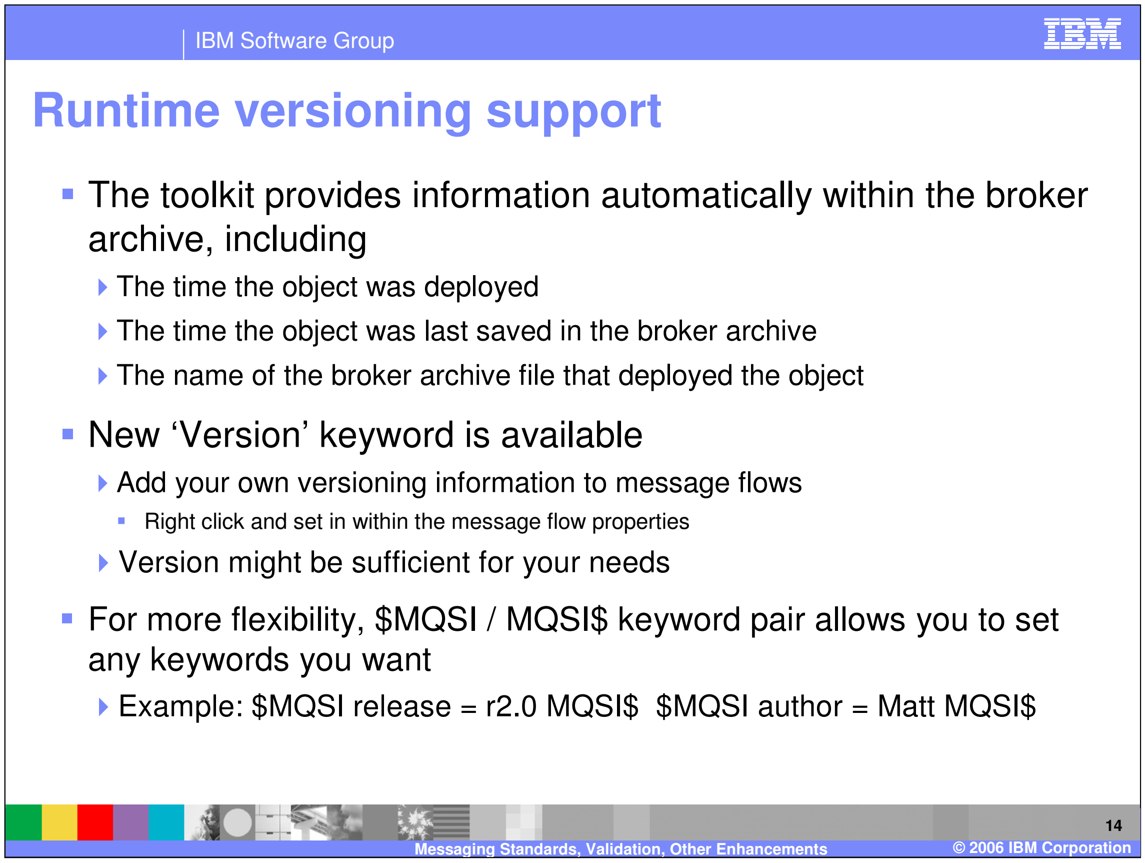
Use MQRFH2C Compact Parser for MQRFH2 Domain: ☐

The “Timing” property in V5 has been moved to the General Message Options in V6, and is called “Parse Timing”. The Parse Timing option can be applied regardless of the master “Validation” property setting. This setting allows complete parsing to be specified independently of other validation settings. The Parse Timing property applies to all domains that support on-demand parsing.

Section

Runtime Versioning

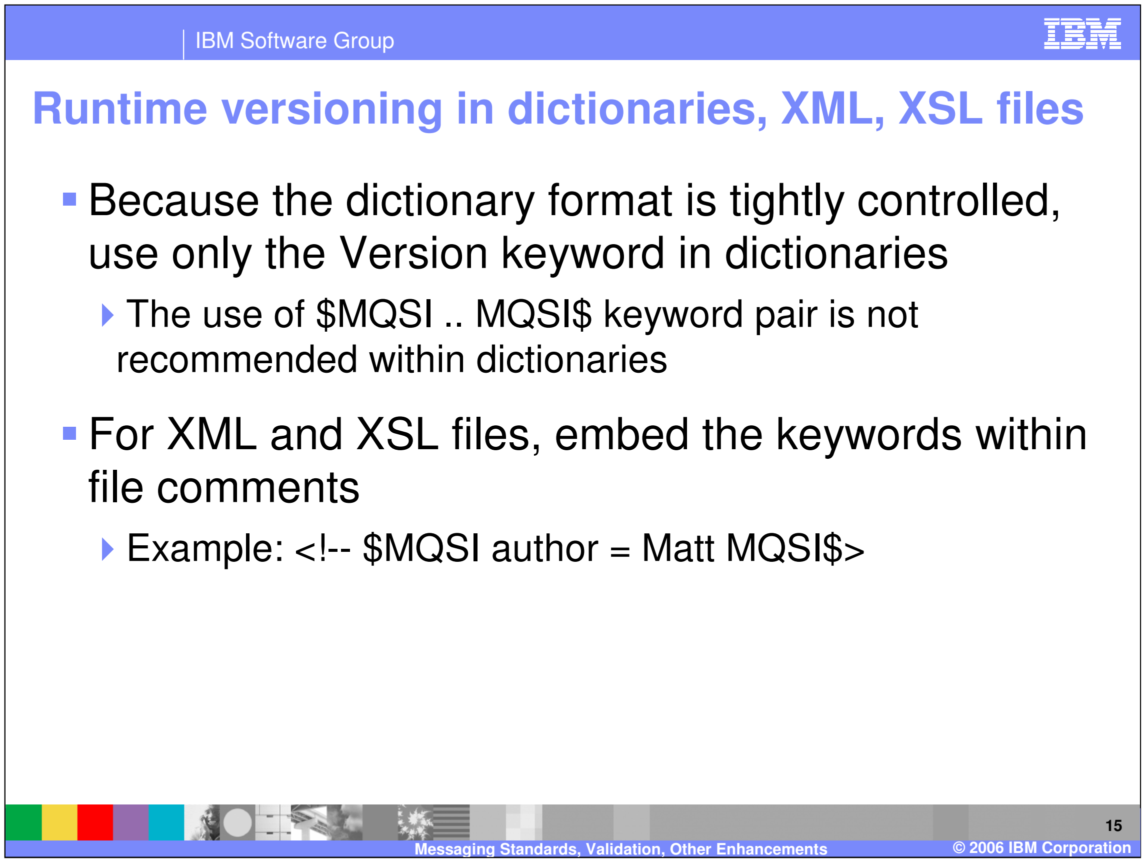
This section discusses the changes in Runtime Versioning.



Runtime versioning support

- The toolkit provides information automatically within the broker archive, including
  - The time the object was deployed
  - The time the object was last saved in the broker archive
  - The name of the broker archive file that deployed the object
- New ‘Version’ keyword is available
  - Add your own versioning information to message flows
    - Right click and set in within the message flow properties
  - Version might be sufficient for your needs
- For more flexibility, $MQSI / MQSI$ keyword pair allows you to set any keywords you want
  - Example: $MQSI release = r2.0 MQSI$ $MQSI author = Matt MQSI$

WebSphere Message Broker V6 provides robust Runtime Versioning support. Some of the functionality for Runtime Versioning is automatically applied during deployment by the Broker Toolkit. For example, when deploying a broker archive file, the toolkit automatically stores the time the object was deployed, the time the object was last saved in the broker archive, and the name of the broker archive file that deployed the object. But in addition to this support, V6 also provides a new “Version” keyword available as a property, which is often sufficient for most versioning requirements. If you have additional requirements for versioning information, you can use the new $MQSI / MQSI$ keyword pair, which allows you the flexibility of adding any additional keyword you want. In the example provided, the new keyword “release” is set to “r2.0”, and it is bounded by the required $MQSI / MQSI$ keyword pair.

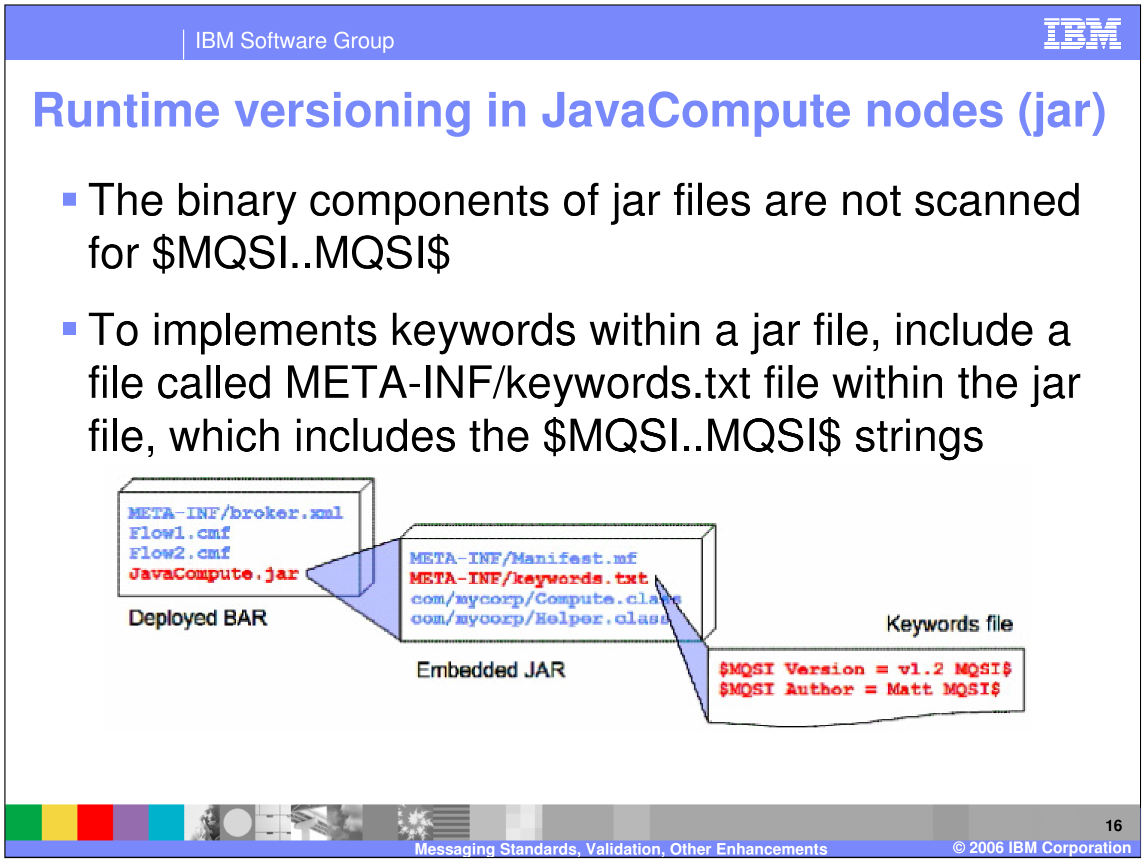


Runtime versioning in dictionaries, XML, XSL files

- Because the dictionary format is tightly controlled, use only the Version keyword in dictionaries
  - The use of $MQSI .. MQSI$ keyword pair is not recommended within dictionaries
- For XML and XSL files, embed the keywords within file comments
  - Example: <!-- $MQSI author = Matt MQSI$>

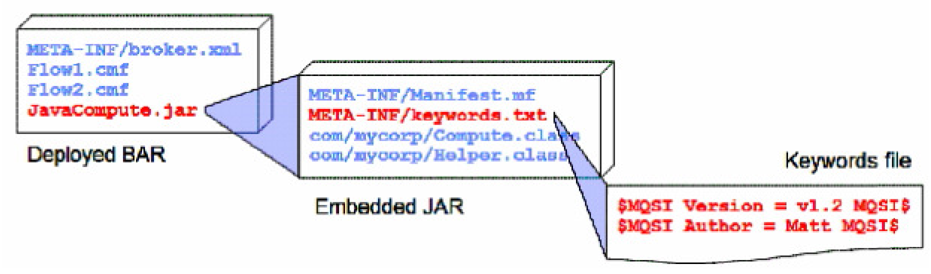
For dictionaries, it is recommended to use only the Version property, since the dictionary format is tightly controlled.

In the case of XML and XSL files, you embed the $MQSI/MQSI$ keyword pair into the files within file comments. In the example provided, a file comment is shown, and within the comment statement the $MQSI/MQSI$ keyword pair encompasses the “author = Matt” keyword and value setting.

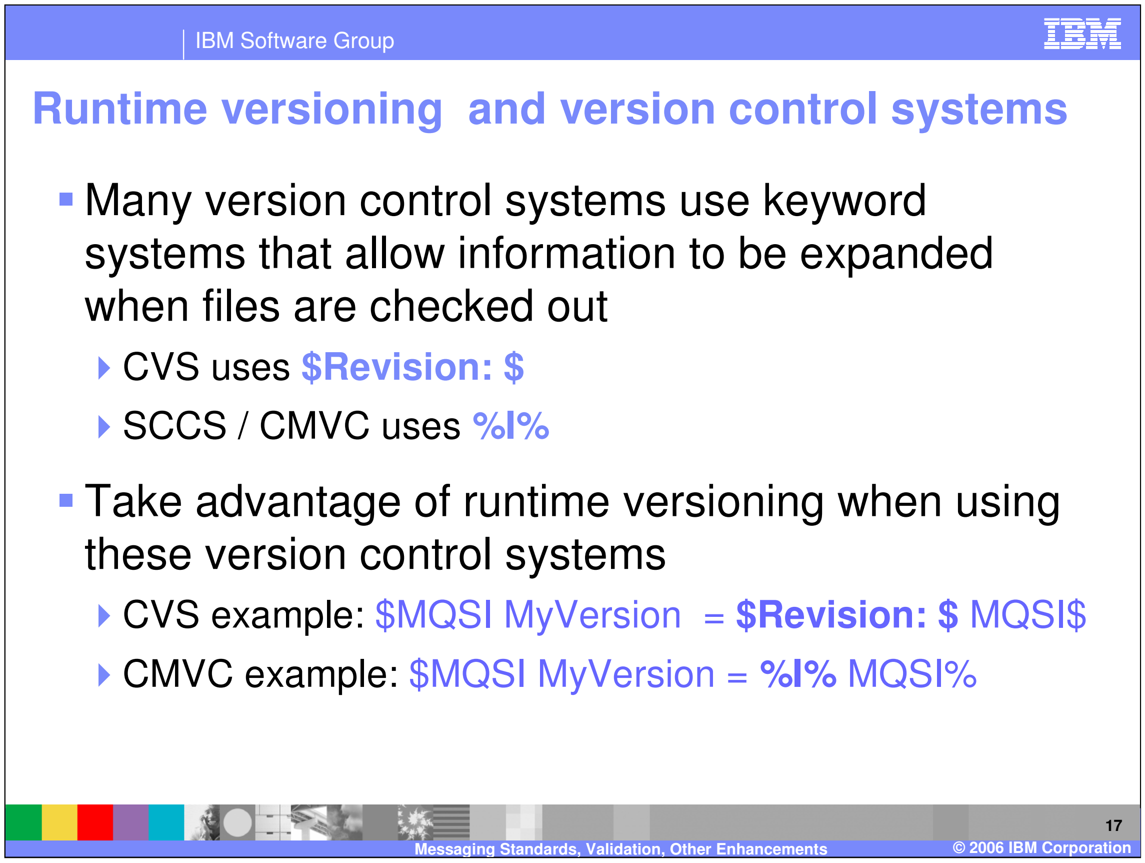


Runtime versioning in JavaCompute nodes (jar)

- The binary components of jar files are not scanned for $MQSI..MQSI$
- To implements keywords within a jar file, include a file called META-INF/keywords.txt file within the jar file, which includes the $MQSI..MQSI$ strings



For the JavaCompute node, which is deployed as a jar file, you cannot add the required keyword pair as elements or comments within the Java™ source, since binary class files are not scanned for the keywords. In this case, you add a file called META-INF/keywords.txt to the jar file, and within this text file you add the $MQSI/MQSI$ keyword pair encompassing the new keywords and values.



Runtime versioning and version control systems

- Many version control systems use keyword systems that allow information to be expanded when files are checked out
  - CVS uses **$Revision: $**
  - SCCS / CMVC uses **%I%**
- Take advantage of runtime versioning when using these version control systems
  - CVS example: $MQSI MyVersion = **$Revision: $** MQSI$
  - CMVC example: $MQSI MyVersion = **%I%** MQSI%

Many version control systems, such as CVS and CMVC, provide special keywords that allow information to be expanded when files are checked out. For example, CVS uses the “dollar sign Revision colon space dollar sign” string to be replaced with CVS version information for that part. For SCCS / CMVC, the string “percent capital I percent” functions the same way. Therefore with these version control systems, you can use these special strings within the $MQSI/MQSI$ keyword pair.

IBM Software Group

Section

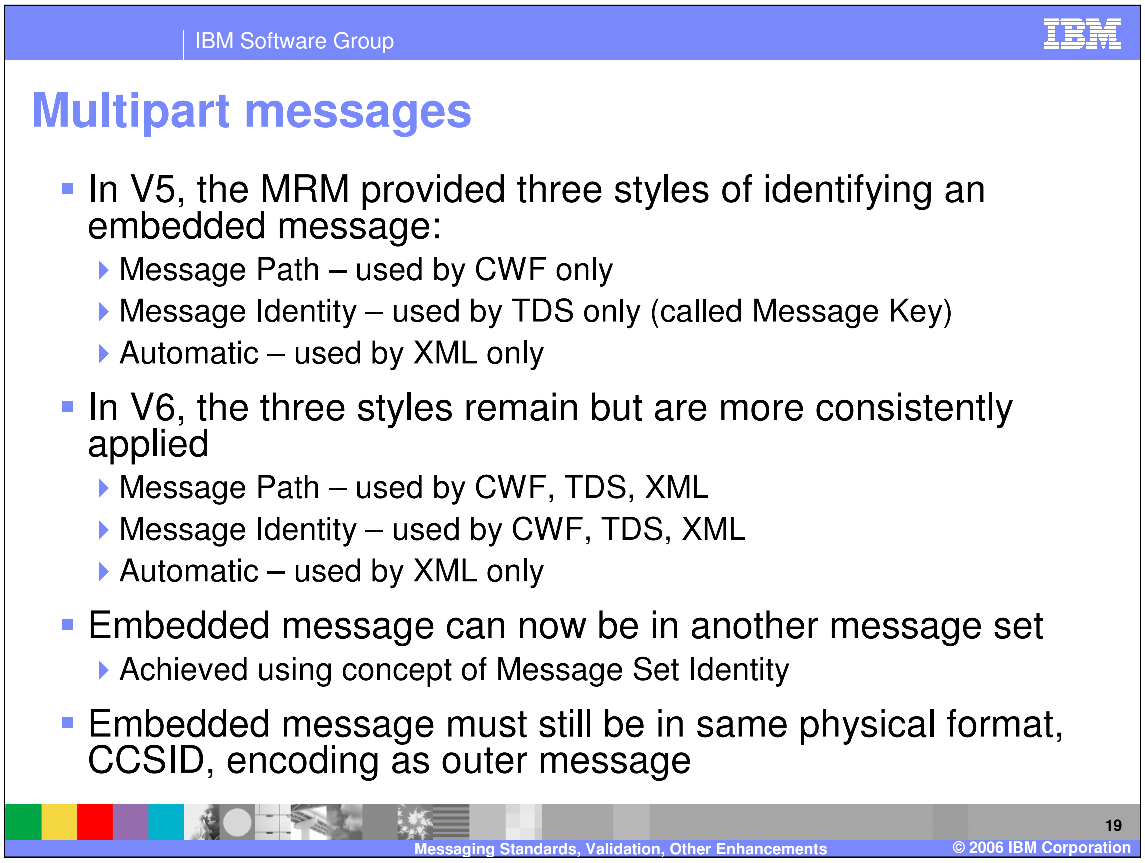
Multipart Messages

18

Messaging Standards, Validation, Other Enhancements

© 2006 IBM Corporation

This section discusses the enhancements to Multipart Messages in V6.



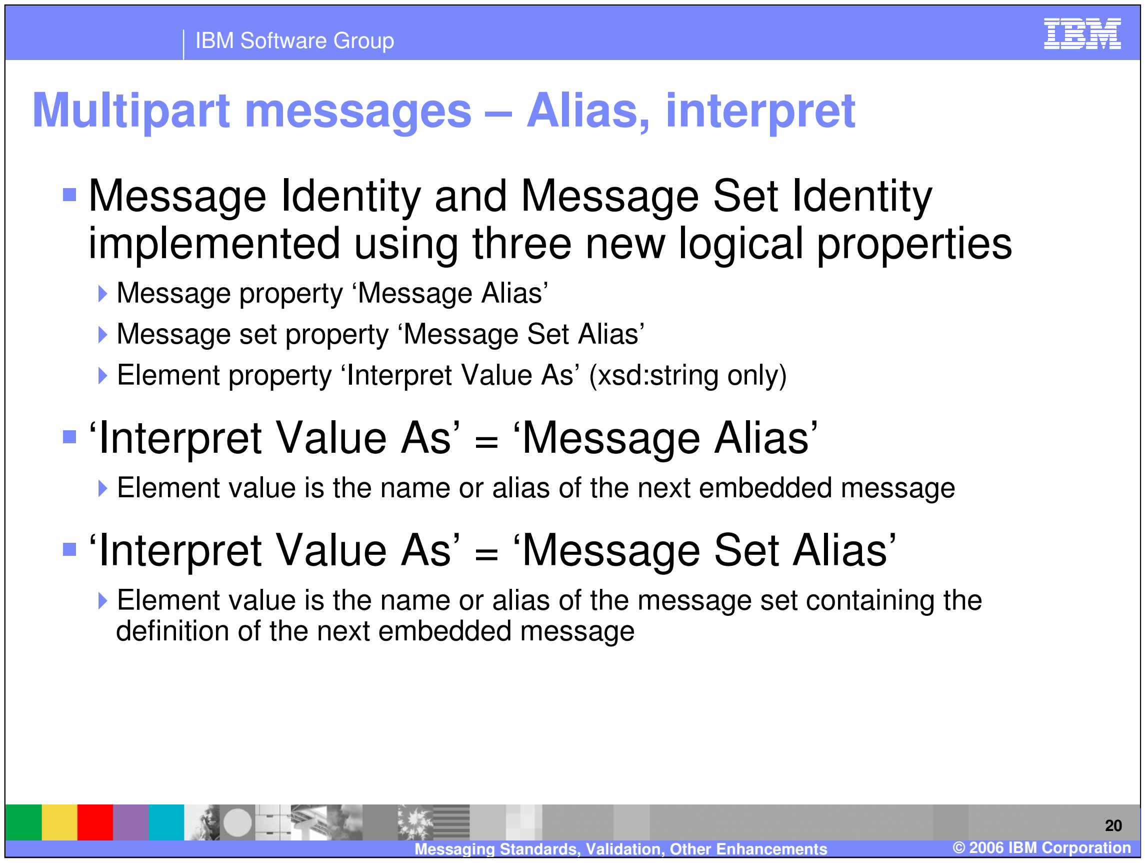
Multipart messages

- In V5, the MRM provided three styles of identifying an embedded message:
  - Message Path – used by CWF only
  - Message Identity – used by TDS only (called Message Key)
  - Automatic – used by XML only
- In V6, the three styles remain but are more consistently applied
  - Message Path – used by CWF, TDS, XML
  - Message Identity – used by CWF, TDS, XML
  - Automatic – used by XML only
- Embedded message can now be in another message set
  - Achieved using concept of Message Set Identity
- Embedded message must still be in same physical format, CCSID, encoding as outer message

In V5, the Message Repository Manager provided three styles of identifying embedded messages: Message Path, which was used by Custom Wire Frame (CWF) only, Message Identity, which was used by Tag Delimited String only, and Automatic, used by XML only.

In V6, those three styles remain but are more consistently applied to the product. Both the Message Path and Message Identity styles are implemented in CWF, TDS, and XML.   The Automatic style remains unchanged in V6.

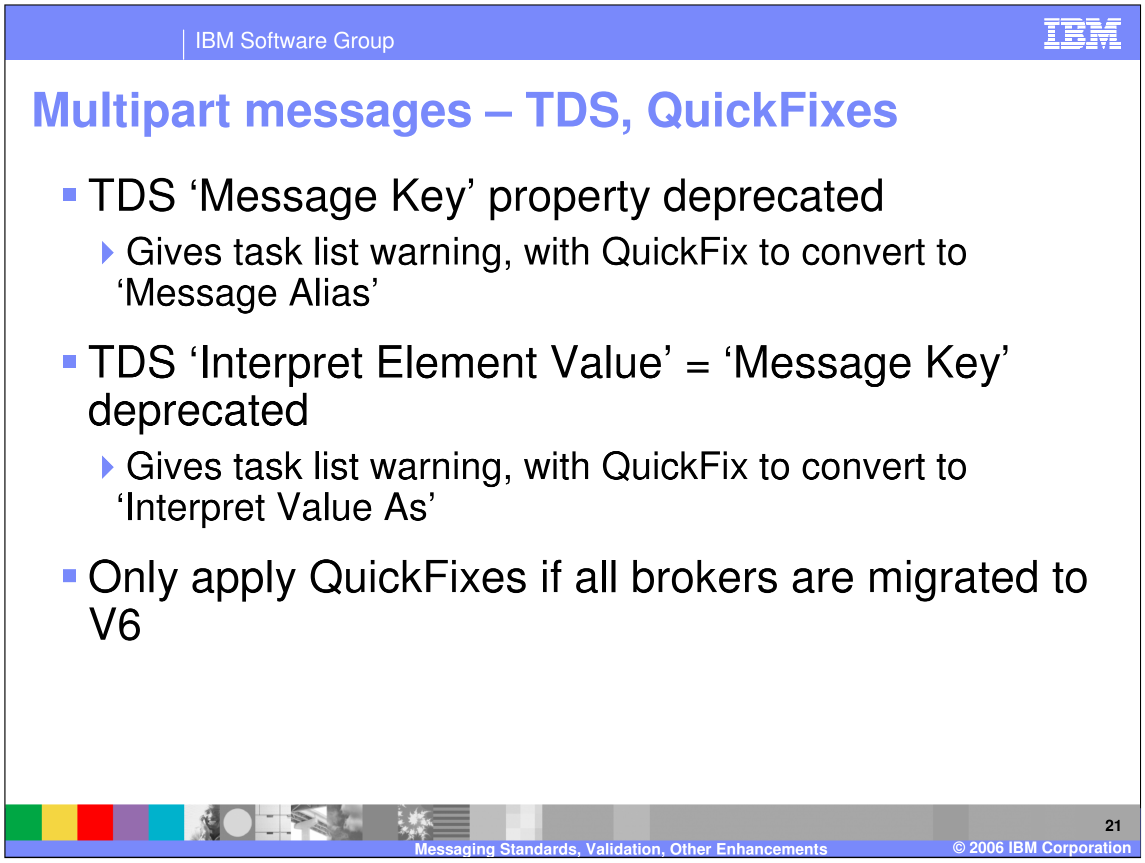
Embedded messages can now be placed in another message set using Message Set Identity. Embedded messages must be in the same physical format, CCSID and encoding as the outer message.



Multipart messages – Alias, interpret

- Message Identity and Message Set Identity implemented using three new logical properties
  - Message property ‘Message Alias’
  - Message set property ‘Message Set Alias’
  - Element property ‘Interpret Value As’ (xsd:string only)
- ‘Interpret Value As’ = ‘Message Alias’
  - Element value is the name or alias of the next embedded message
- ‘Interpret Value As’ = ‘Message Set Alias’
  - Element value is the name or alias of the message set containing the definition of the next embedded message

Embedded messages using the Message Identity or Message Set Identity techniques have three new logical properties which can be specified in V6: Message Alias, Message Set Alias, and the “Interpret Value As” element property. For cases where the Message Identity element value does not match the *Name* of the message, you should use the *Message Alias* property to specify this value. The MRM parser will try to match on *Name* first and if that fails it will try to match on *Message Alias*. For cases where the Message Set Identity element value does not match the *Identifier* or *Name* of the message set, you should use the *Message Set Alias* property to specify this value. The MRM parser will try to match on *Identifier* first, then on *Name*, and finally *Message Set Alias*.



Multipart messages – TDS, QuickFixes

- TDS ‘Message Key’ property deprecated
  - Gives task list warning, with QuickFix to convert to ‘Message Alias’
- TDS ‘Interpret Element Value’ = ‘Message Key’ deprecated
  - Gives task list warning, with QuickFix to convert to ‘Interpret Value As’
- Only apply QuickFixes if all brokers are migrated to V6

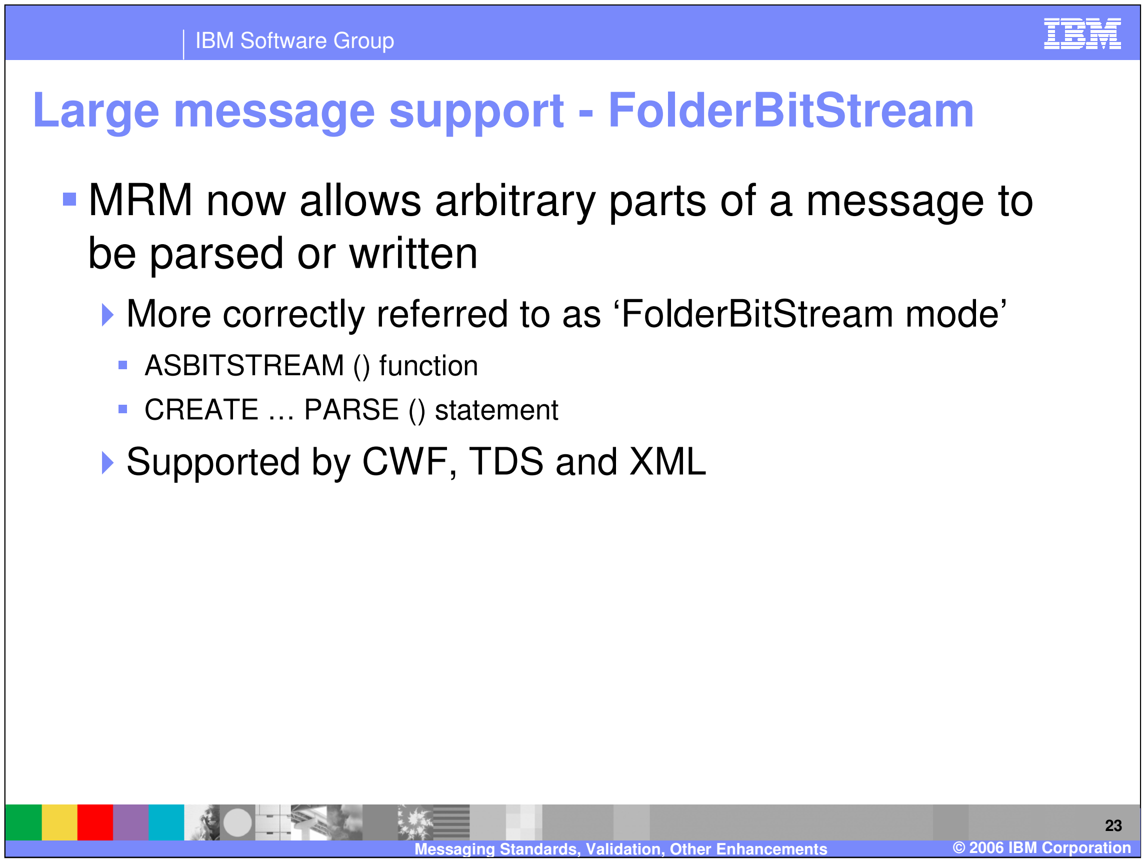
In regard to Multipart Messages and TDS, be aware that versions of the TDS physical format prior to Version 6.0 included embedded message identification by Message Key which worked in a similar manner to Message Identity, but which applied to TDS only. The Message Key technique has been deprecated and is superseded by Message Identity. Warning task list messages are issued if the use of Message Key is detected, and a task list Quick Fix may be selected to create the equivalent Message Identity automatically. The same is true for the deprecated TDS “Interpret Element Value”, which should be converted to “Interpret Value As”, either manually or with QuickFix.

Note that QuickFixes should only be used if all the brokers are migrated to Version 6.

Section

Other enhancements

This section will discuss additional enhancements.



Large message support - FolderBitStream

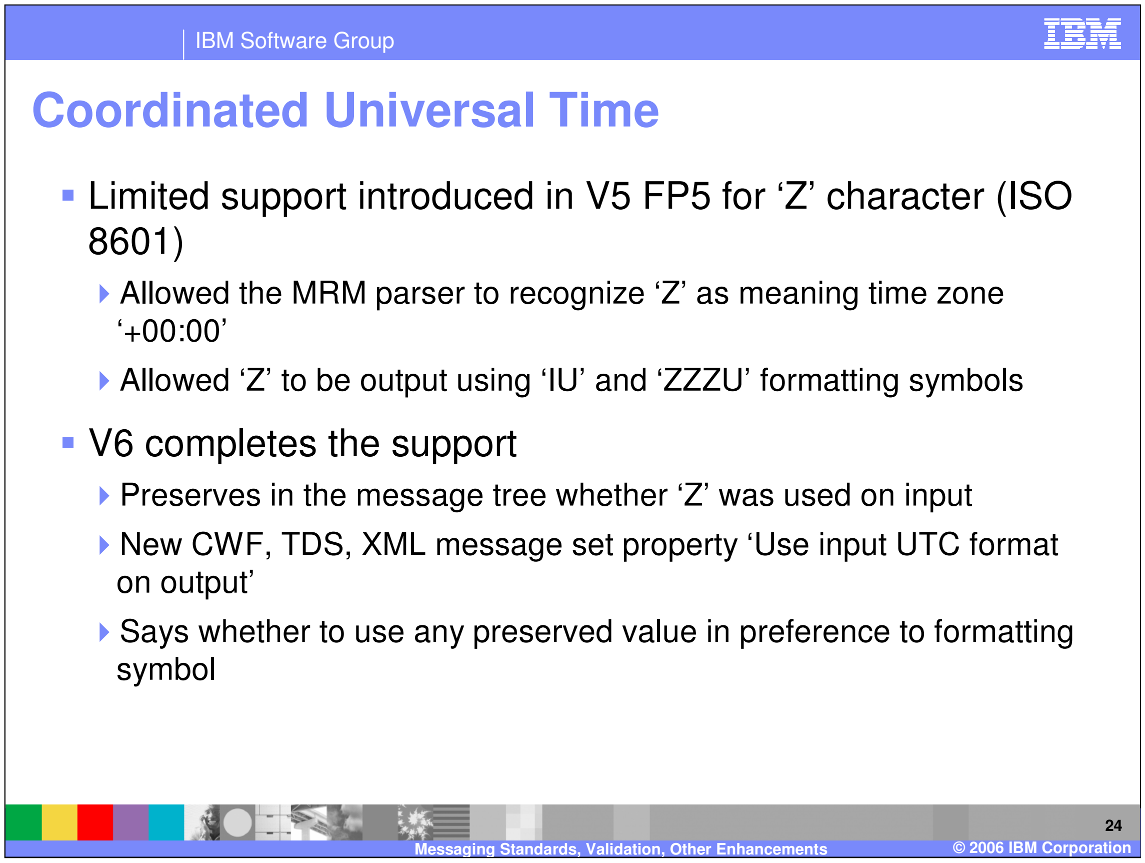
- MRM now allows arbitrary parts of a message to be parsed or written
  - More correctly referred to as ‘FolderBitStream mode’
    - ASBITSTREAM () function
    - CREATE … PARSE () statement
  - Supported by CWF, TDS and XML

WebSphere Message Broker V6 now allows arbitrary parts of a message to be individually parsed or written. Specifically, this new functionality can be used with ASBITSTREAM and with “CREATE … PARSE()”.

If you code the ASBITSTREAM function with the parser mode option set to *FolderBitStream*, to parse a message tree to a bit stream, the generated bit stream is an MRM element built from the target element and its children. Unlike *RootBitStream* mode the target element does not have to represent a message; it can represent a predefined element within a message or self-defined element within a message. So that the MRM parser can correctly parse the message, the path from the message to the target element within the message must be specified in the *Message Type*. The format of the path is the same as that used by message paths except that the message type prefix is not used.

If you code a CREATE statement with a PARSE clause, with the parser mode option set to *FolderBitStream*, to parse a bit stream to a message tree, the expected bit stream is a document in the format specified by the Message Format, which is either specified directly or inherited. Unlike *RootBitStream* mode the root of the document does not have to represent an MRM message; it can represent a predefined element within a message or self-defined element within a message.

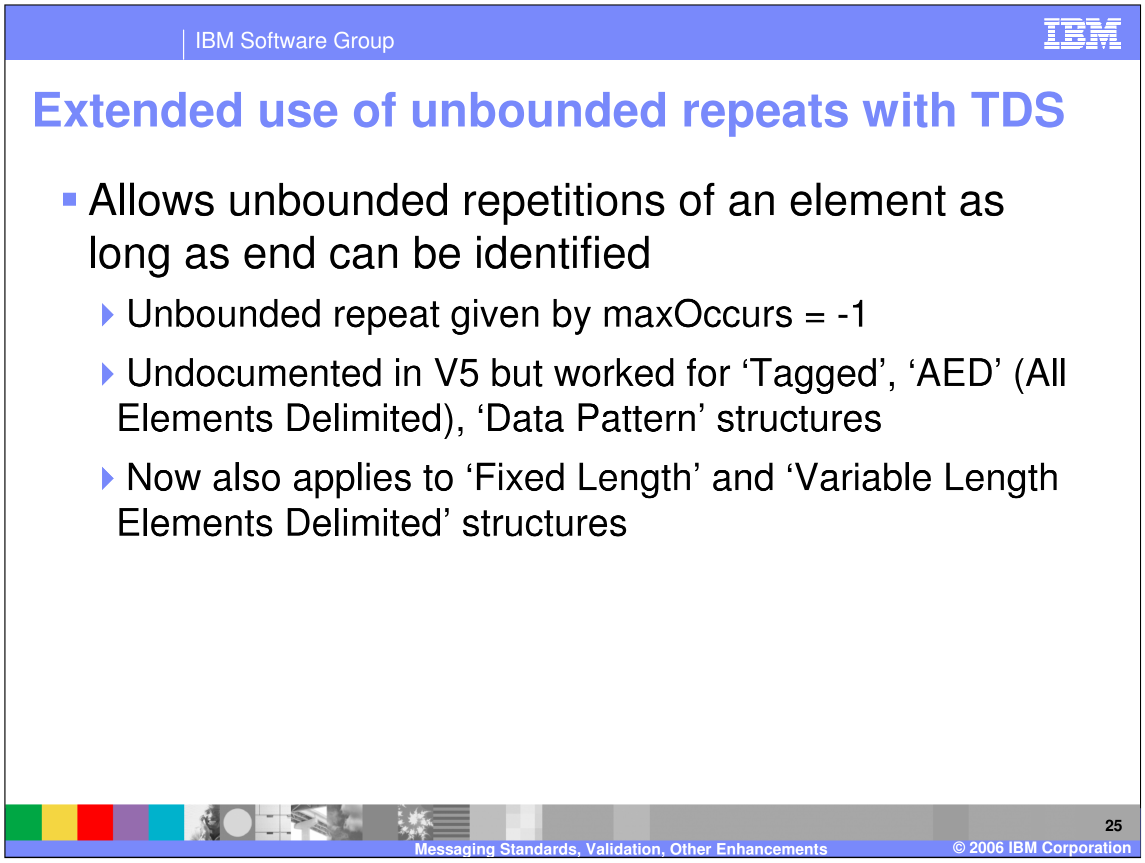
AsBitStream and Create..Parse() are supported by CWF, TDS, and XML formats.



Coordinated Universal Time

- Limited support introduced in V5 FP5 for ‘Z’ character (ISO 8601)
  - Allowed the MRM parser to recognize ‘Z’ as meaning time zone ‘+00:00’
  - Allowed ‘Z’ to be output using ‘IU’ and ‘ZZZU’ formatting symbols
- V6 completes the support
  - Preserves in the message tree whether ‘Z’ was used on input
  - New CWF, TDS, XML message set property ‘Use input UTC format on output’
  - Says whether to use any preserved value in preference to formatting symbol

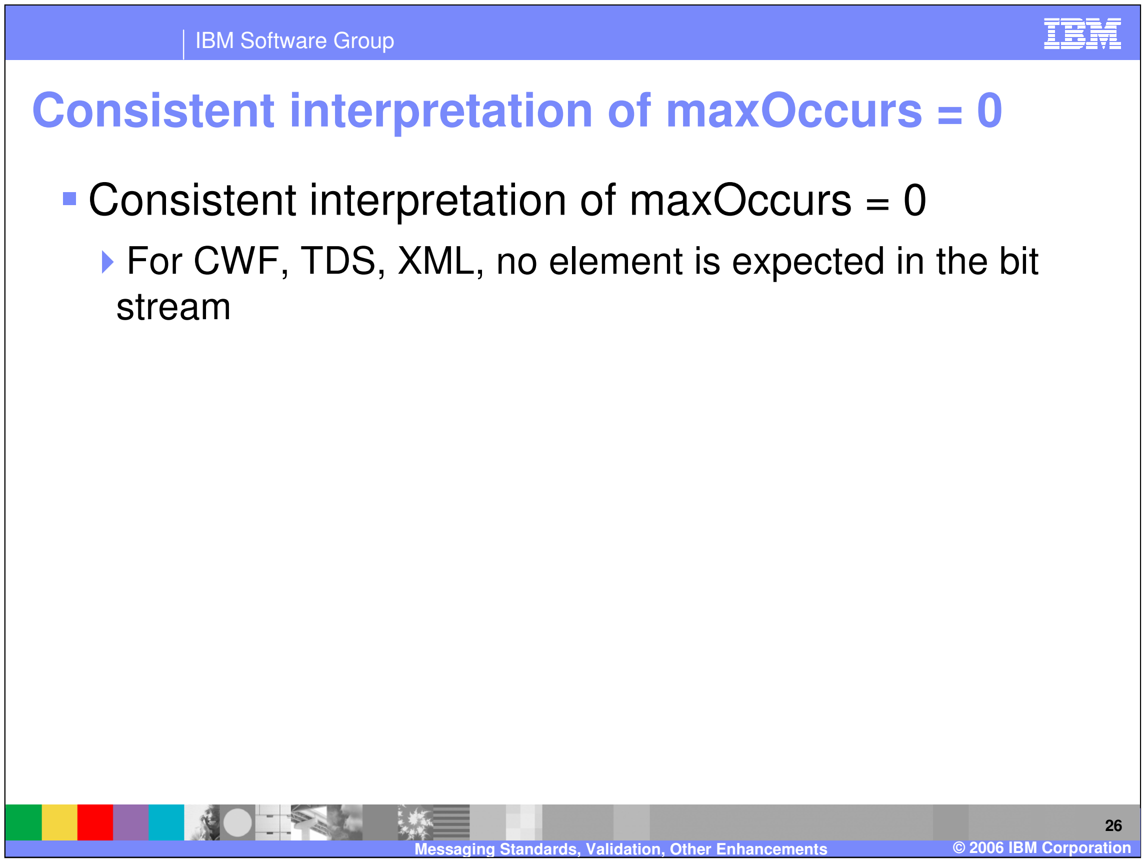
While V5 had limited support for ISO 8601 (International Standard Date and Time Notation), V6 completes the support for this standard. V6 preserves the time zone setting that was provided by the input within the message tree. In addition, it now provides the appropriate message set properties relating to UTC (Coordinated Universal Time). Finally, V6 has a property to indicate whether or not the input formatting value will be preserved into the output message, overriding the DateTime format property.



Extended use of unbounded repeats with TDS

- Allows unbounded repetitions of an element as long as end can be identified
  - Unbounded repeat given by maxOccurs = -1
  - Undocumented in V5 but worked for 'Tagged', 'AED' (All Elements Delimited), 'Data Pattern' structures
  - Now also applies to 'Fixed Length' and 'Variable Length Elements Delimited' structures

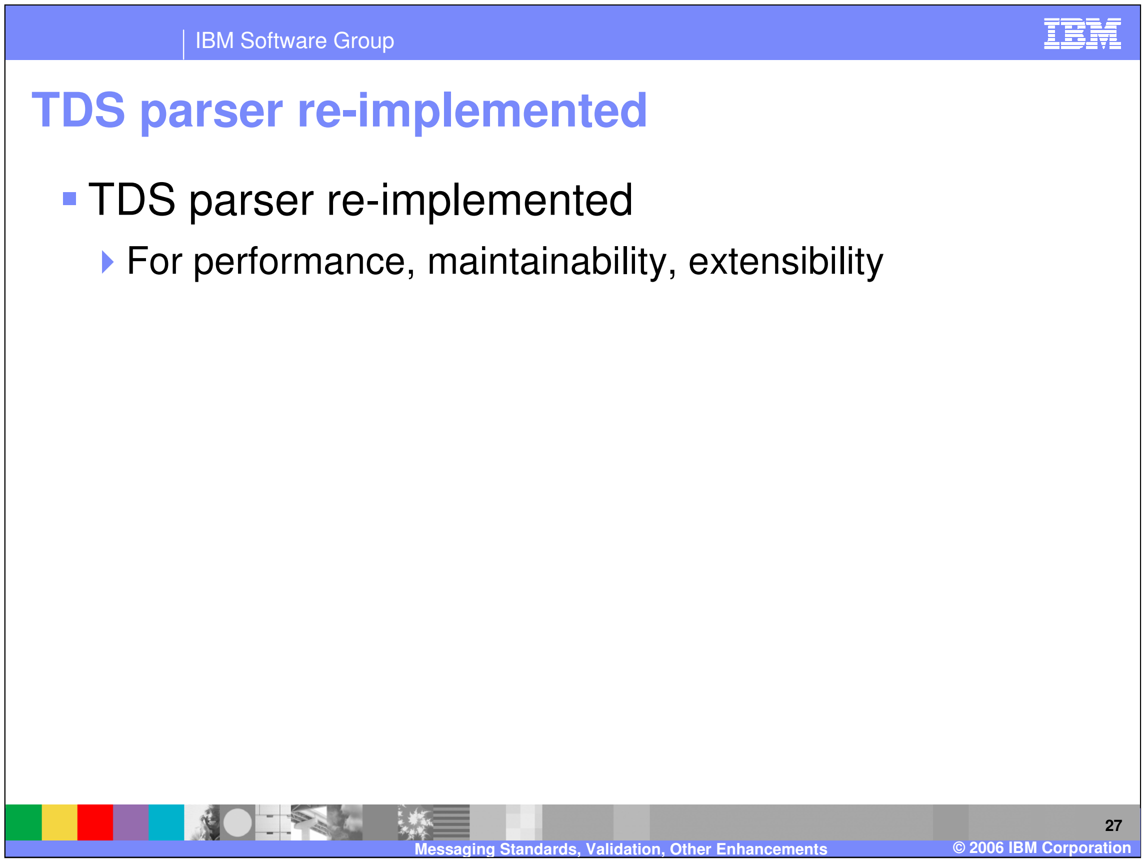
V6 allows the TDS format structure to have unbounded repeats of an element using the setting "maxOccurs = -1". In V6, this feature applies to Tagged, AED, Data Pattern, Fixed Length, and Variable Length Delimited structures.



Consistent interpretation of maxOccurs = 0

- Consistent interpretation of maxOccurs = 0
  - For CWF, TDS, XML, no element is expected in the bit stream

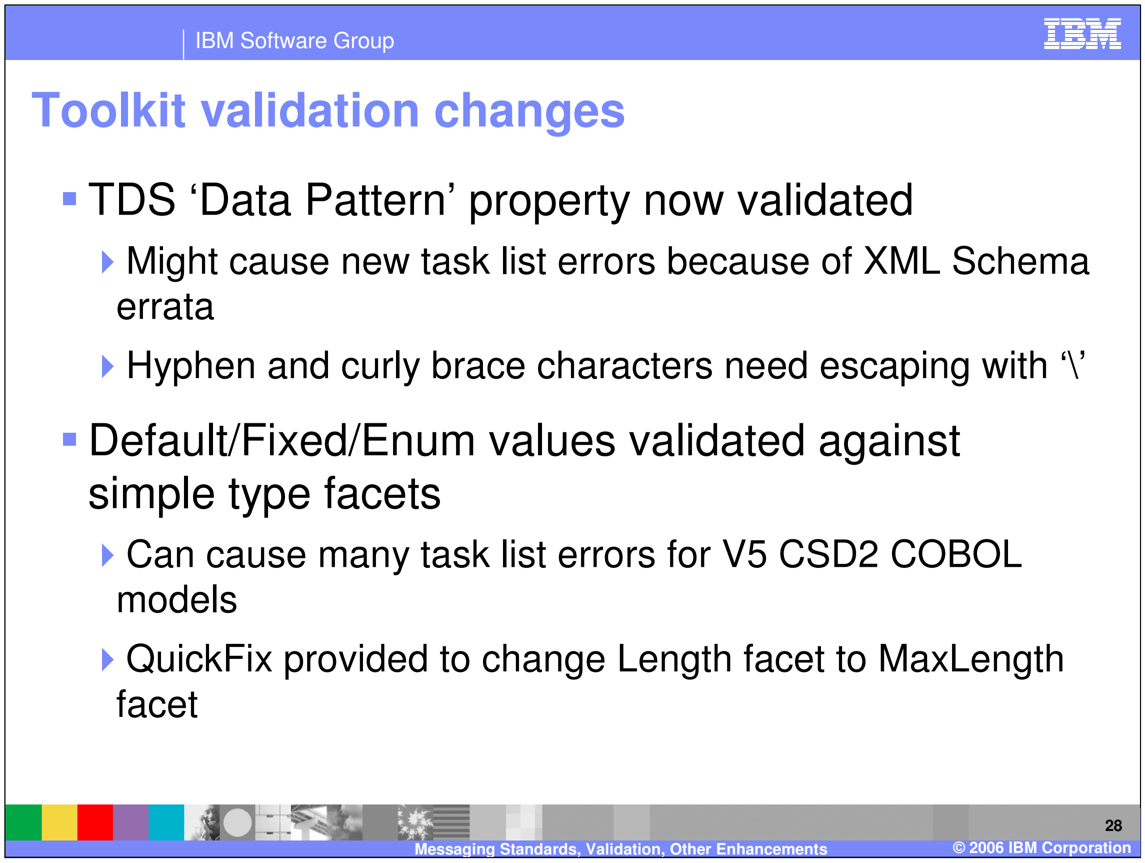
In V6, within the CWF, TDS, and XML formats, maxOccurs=0 is taken to strictly mean that no element is expected in the bit stream.



TDS parser re-implemented

- TDS parser re-implemented
  - For performance, maintainability, extensibility

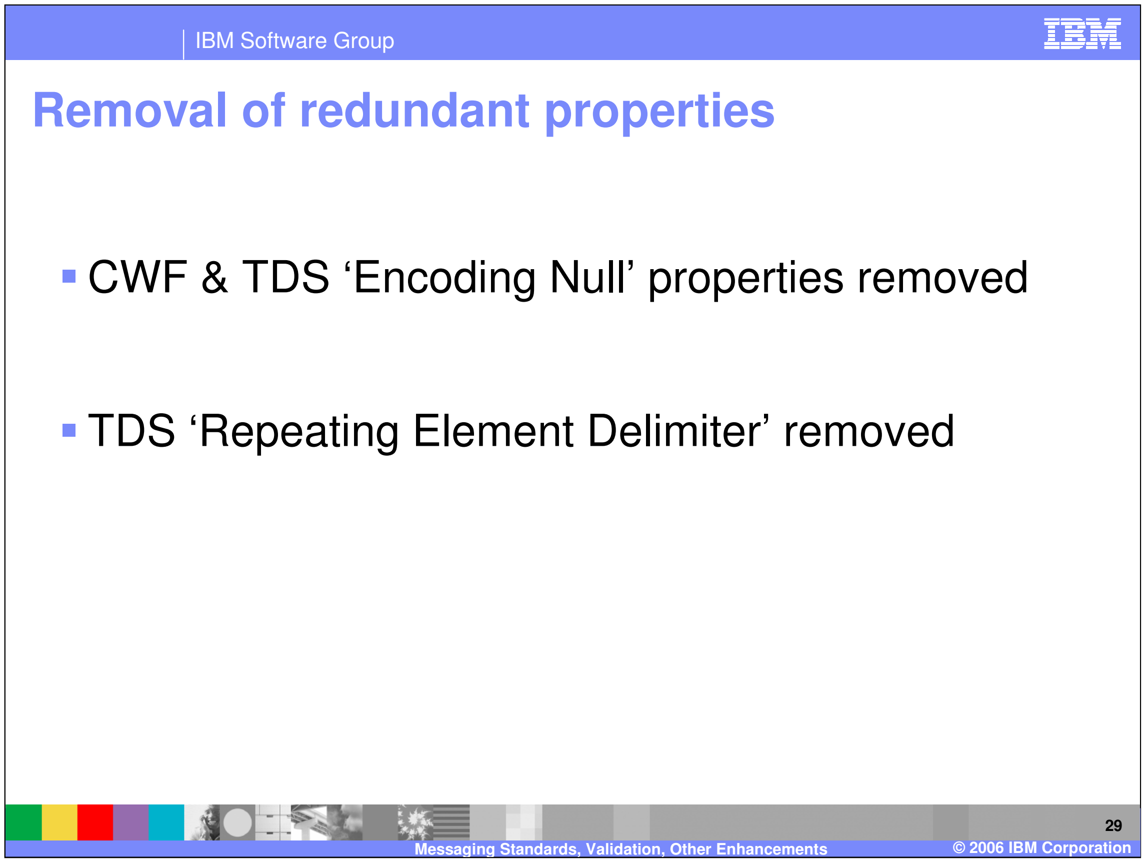
The TDS parser has been changed in V6 to provide better performance, maintainability, and extensibility than in V5.



Toolkit validation changes

- TDS ‘Data Pattern’ property now validated
  - Might cause new task list errors because of XML Schema errata
  - Hyphen and curly brace characters need escaping with ‘\’
- Default/Fixed/Enum values validated against simple type facets
  - Can cause many task list errors for V5 CSD2 COBOL models
  - QuickFix provided to change Length facet to MaxLength facet

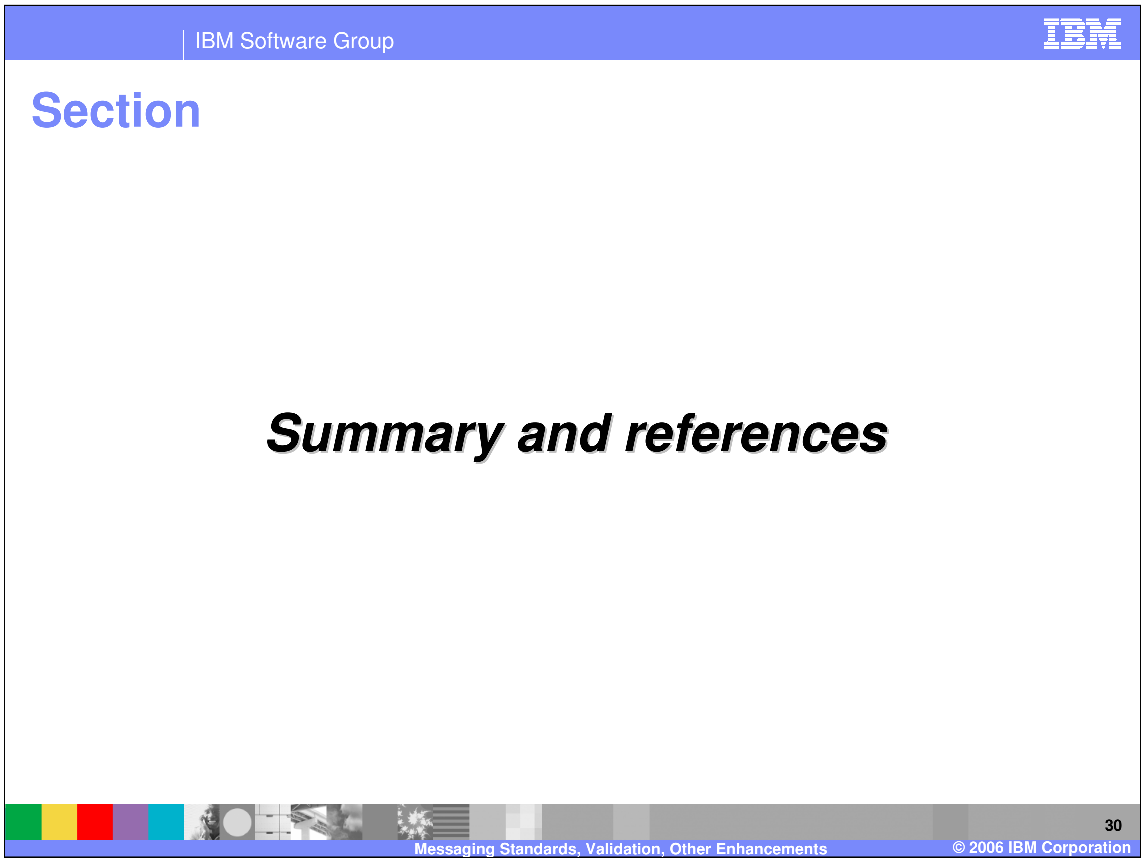
In V6, additional Toolkit validations are performed, which may uncover some new task list errors because of XML Schema errata. For example, the hyphen and curly brace characters need escaping by a preceding backslash (\). The “Default/Fixed/Enum” values are now validated against simple type facets. For this case, consider using the QuickFix to change the Length facet to MaxLength facet.



Removal of redundant properties

- CWF & TDS ‘Encoding Null’ properties removed
- TDS ‘Repeating Element Delimiter’ removed

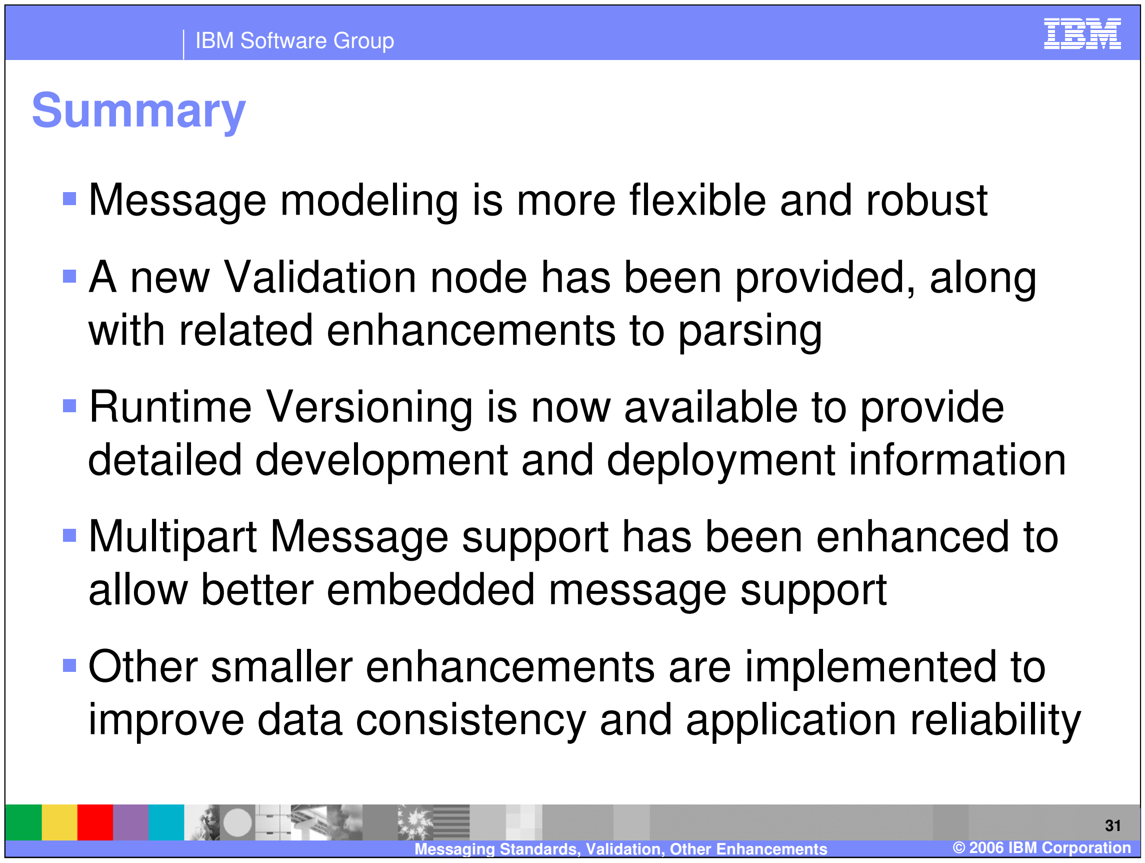
Certain redundant properties have been removed in V6. The CWF and TDS “Encoding Null” property has been removed from the properties. In addition, the TDS “Repeating Element Delimiter” has been removed from the properties.



Section

Summary and references

The last portion of the presentation contains a summary and references.



Summary

- Message modeling is more flexible and robust
- A new Validation node has been provided, along with related enhancements to parsing
- Runtime Versioning is now available to provide detailed development and deployment information
- Multipart Message support has been enhanced to allow better embedded message support
- Other smaller enhancements are implemented to improve data consistency and application reliability

In summary, V6 provides more flexible and robust message modeling. A new Validation node is available allowing for greater message content quality and control, along with related enhancements to parsing. Runtime Versioning allows you to gather, maintain, and display detailed development and deployment information. Multipart Messaging support is enhanced with new functionality, particularly in the area of embedded messages. And finally, a number of smaller refinements in the area of Message Modeling have been made to improve data consistency and application reliability.

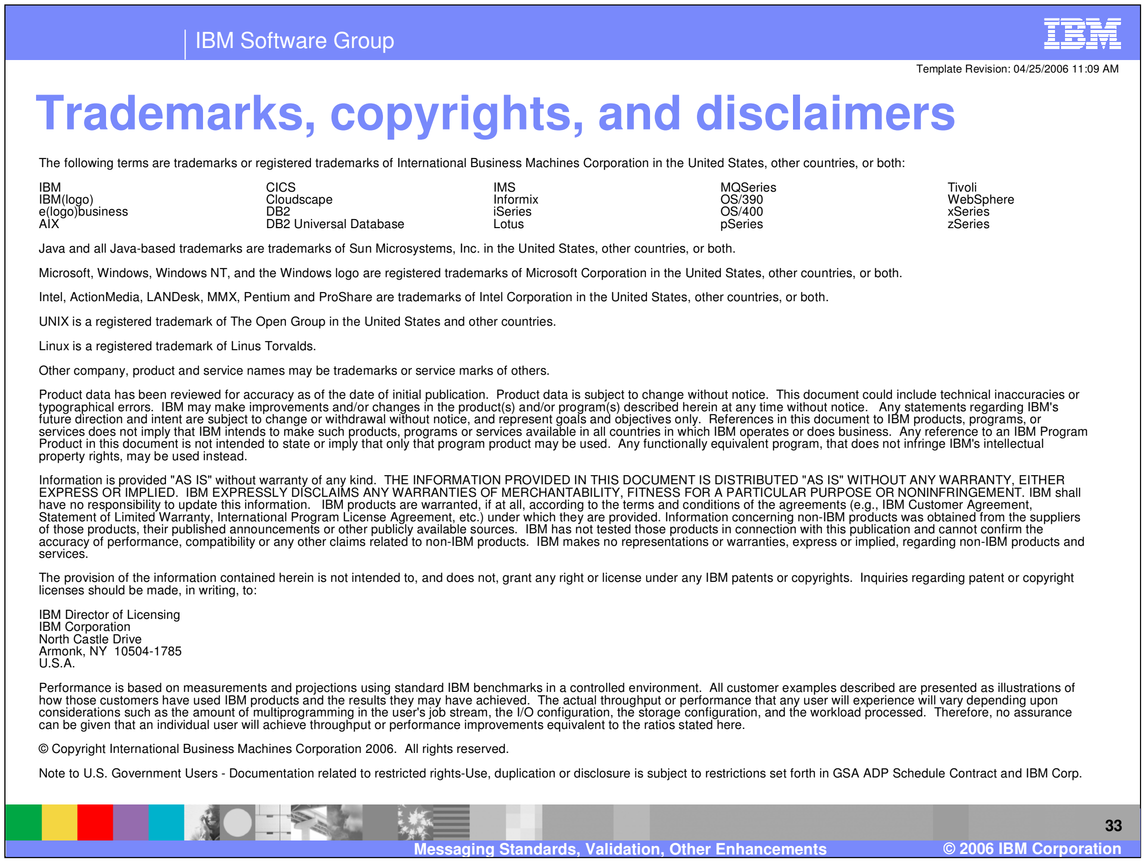
References

- WebSphere Message Broker library:

http://www-306.ibm.com/software/integration/wbimessagebroker/library/

- WebSphere Message Broker Information Center:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp



Trademarks, copyrights, and disclaimers