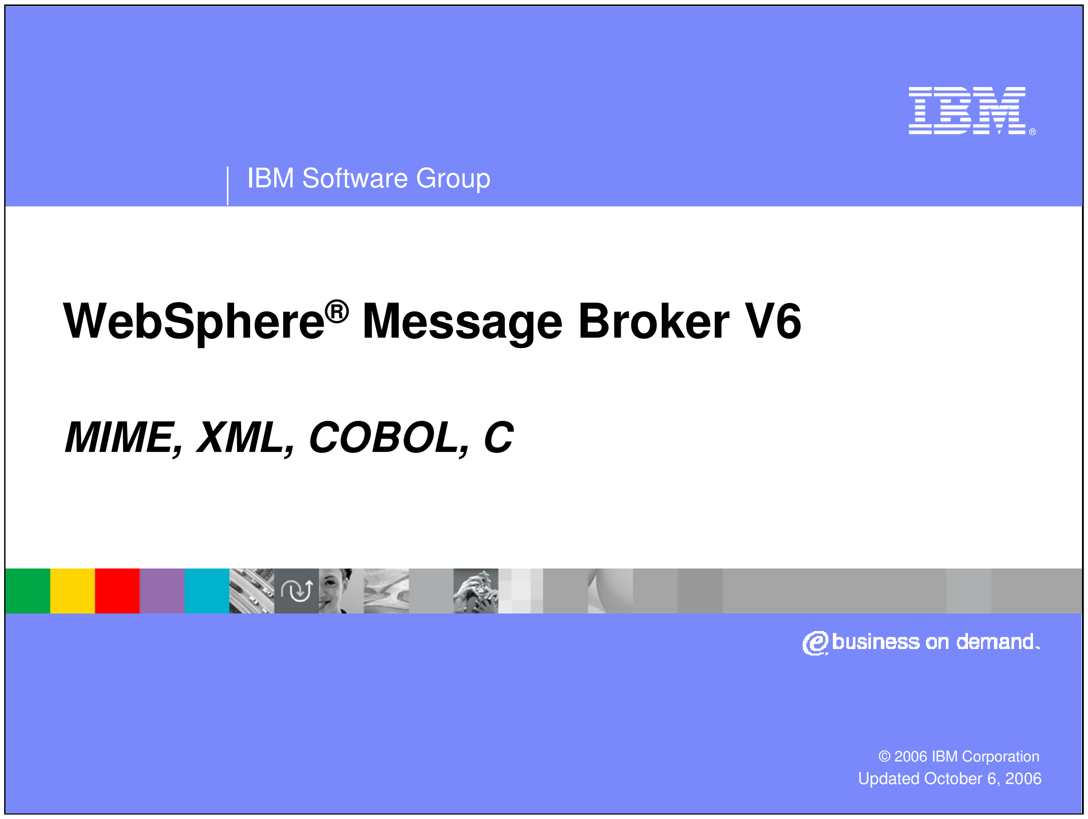


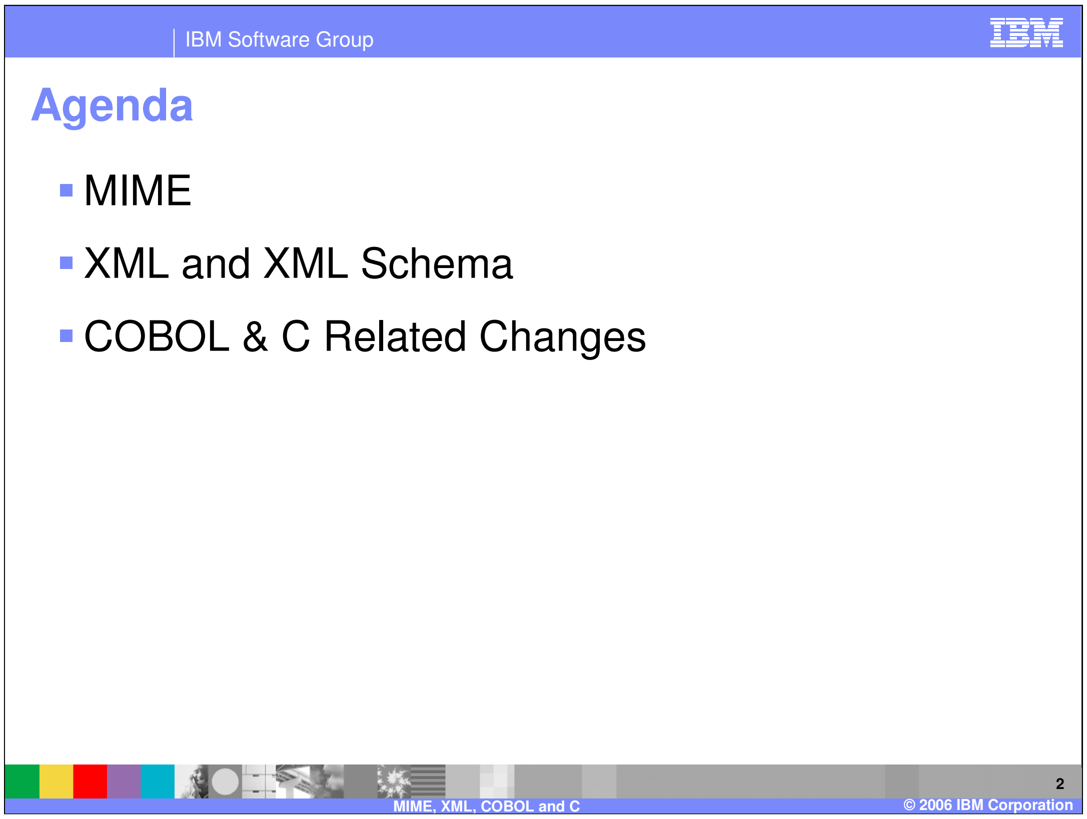
IBM Software Group

WebSphere® Message Broker V6

MIME, XML, COBOL, C

business on demand.

This presentation discusses changes in MIME, XML, COBOL and C support in WebSphere Message Broker V6.



Agenda

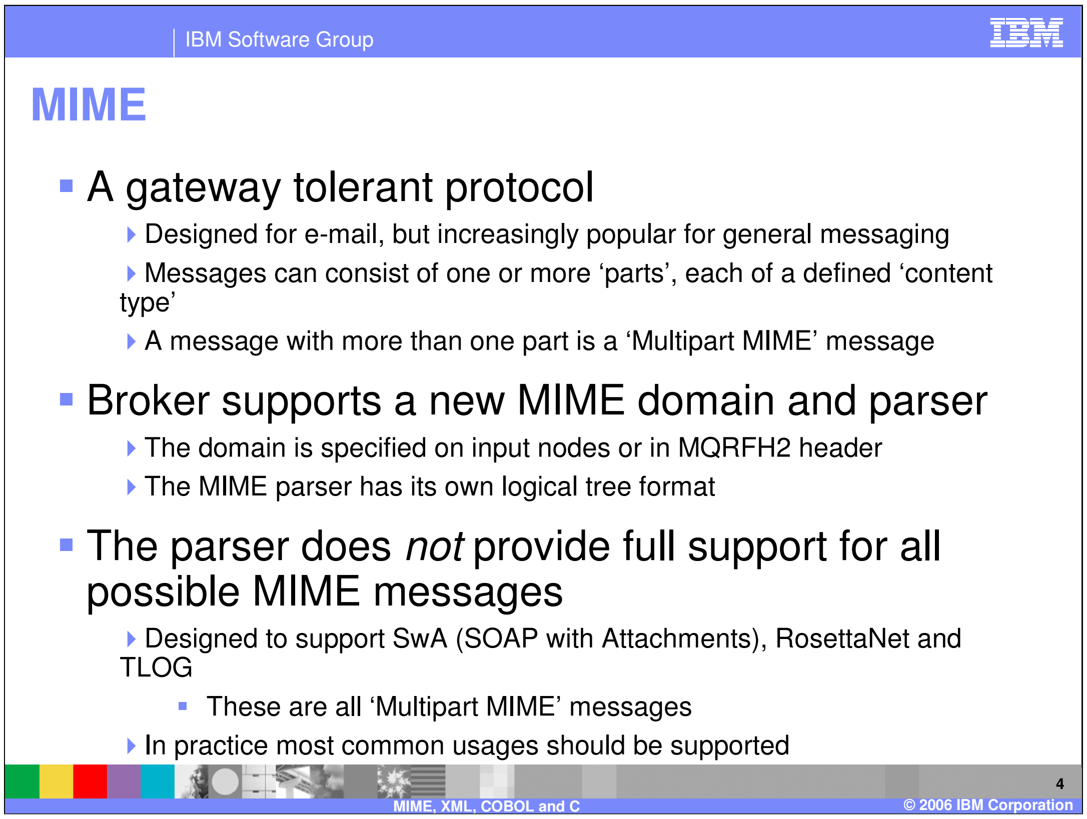
- MIME
- XML and XML Schema
- COBOL & C Related Changes

The topics presented in this presentation include MIME, XML & XML Schema, COBOL and C related changes.

Section

MIME

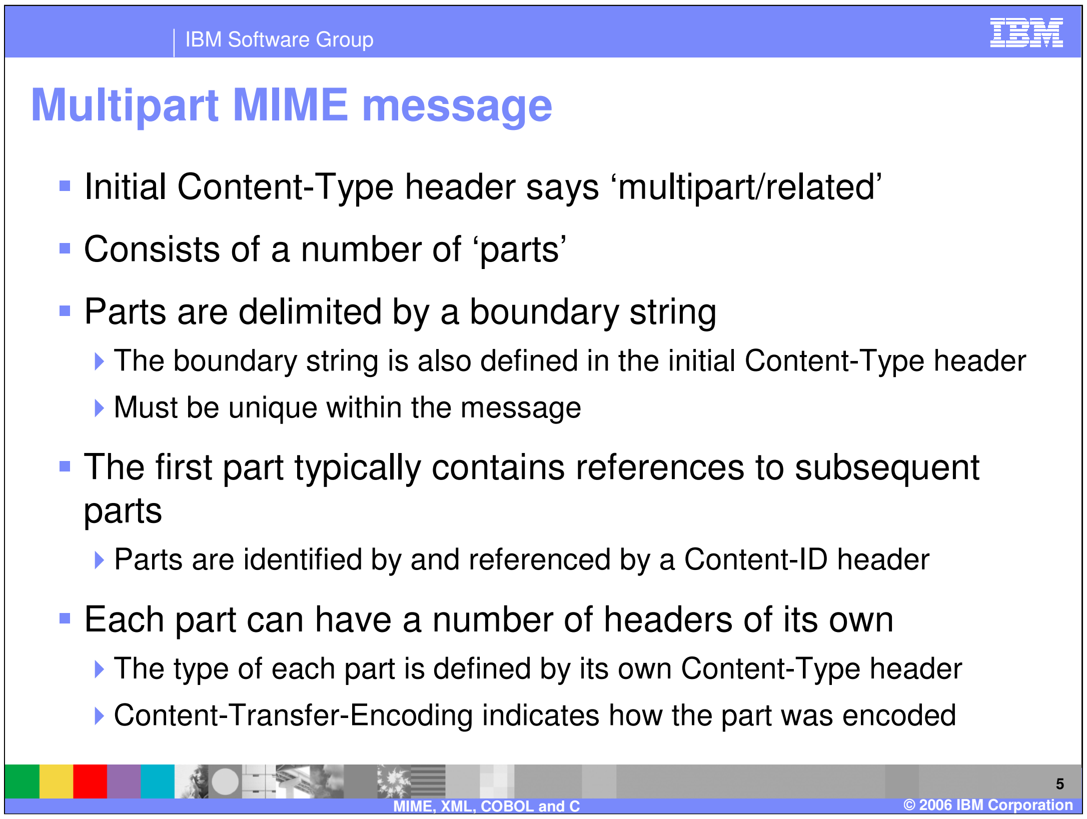
The MIME domain should be used if your messages exploit the MIME standard for multipart messages. The following slides summarize the enhancements to the MIME domain support in WebSphere Message Broker V6.



MIME

- A gateway tolerant protocol
  - Designed for e-mail, but increasingly popular for general messaging
  - 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
- Broker supports a new MIME domain and parser
  - The domain is specified on input nodes or in MQRFH2 header
  - The MIME parser has its own logical tree format
- The parser does *not* provide full support for all possible MIME messages
  - Designed to support SwA (SOAP with Attachments), RosettaNet and TLOG
    - These are all ‘Multipart MIME’ messages
  - In practice most common usages should be supported

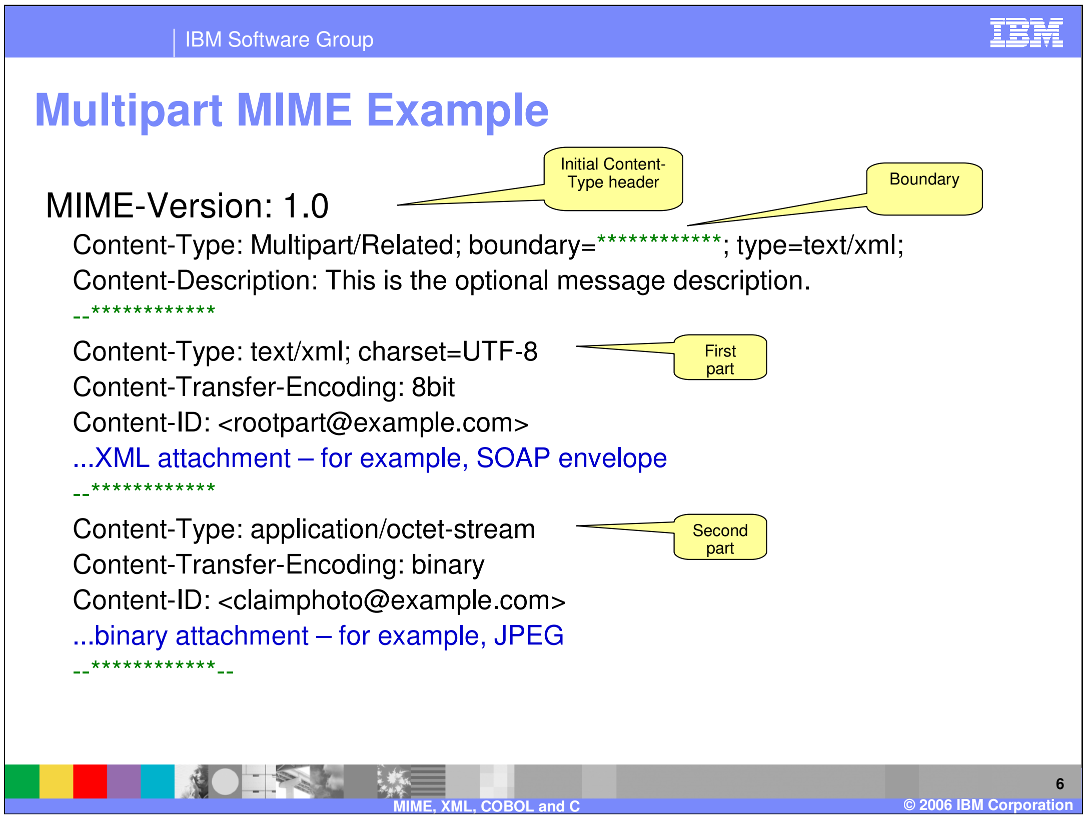
MIME or Multipurpose Internet Mail Extensions, was originally designed for e-mail but has become increasingly popular for general messaging. This standard allows for messages to consist of one or more parts, with each part identified with a “content type”. The MIME domain and parser implementations in V6 are rewritten to provide greater function and flexibility. A MIME message of more than one part is called a “multipart MIME message”. The domain name is specified on input nodes or in MQRFH2 headers. The MIME parser creates its own logical tree using bit stream metadata. The MIME domain that is implemented in WebSphere Message Broker V6 does not support the full MIME standard, but it does support the common MIME formats in use in message-based applications, including SOAP with Attachments (or SwA), RosettaNet and TLOG. In practice you will find that most usages should be supported.



Multipart MIME message

- Initial Content-Type header says ‘multipart/related’
- Consists of a number of ‘parts’
- Parts are delimited by a boundary string
  - The boundary string is also defined in the initial Content-Type header
  - Must be unique within the message
- The first part typically contains references to subsequent parts
  - Parts are identified by and referenced by a Content-ID header
- Each part can have a number of headers of its own
  - The type of each part is defined by its own Content-Type header
  - Content-Transfer-Encoding indicates how the part was encoded

A multipart MIME message contains an initial Content-Type header which indicates ‘multipart/related’ followed by a number of parts, delimited by a unique boundary string defined in the initial ‘Content-Type’ header. Each subsequent message part also contains its own ‘Content-ID’ header, with a ‘Content-Transfer-Encoding’ property to indicate how that particular part was encoded.

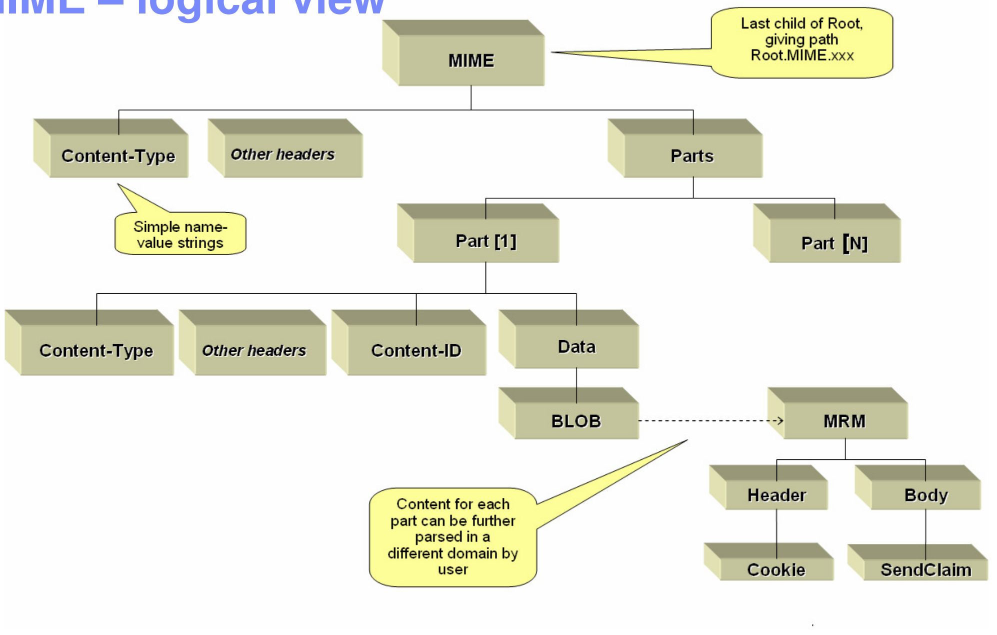


Multipart MIME Example

```
MIME-Version: 1.0
   Content-Type: Multipart/Related; boundary=************; type=text/xml;
   Content-Description: This is the optional message description.
   --************
   Content-Type: text/xml; charset=UTF-8
   Content-Transfer-Encoding: 8bit
   Content-ID: <rootpart@example.com>
   ...XML attachment – for example, SOAP envelope
   --************
   Content-Type: application/octet-stream
   Content-Transfer-Encoding: binary
   Content-ID: <claimphoto@example.com>
   ...binary attachment – for example, JPEG
   --************--
```

Here is a simple example of a MIME multipart message. The initial ‘Content-Type’ indicates that the message is a multipart message of related messages, and defines the boundary string using a set of characters that must be unique within the message. The following two message parts shown also contain their own ‘Content-Type’ headers, ‘Content-Transfer-Encoding’ settings, and ‘Content-ID’ settings.

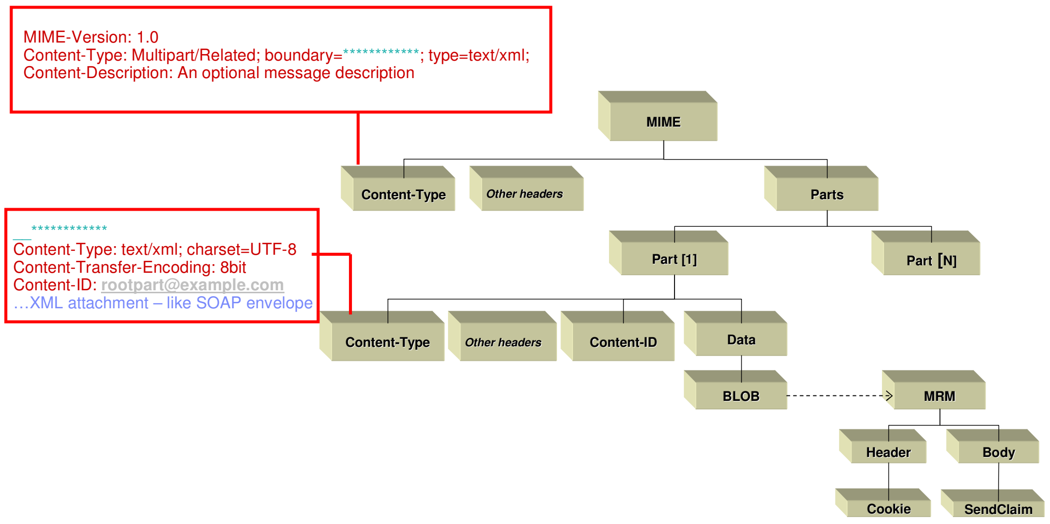
MIME – logical view


MIME
Last child of Root, giving path Root.MIME.xxx
Content-Type
Other headers
Parts
Simple name-value strings
Part [1]
Part [N]
Content-Type
Other headers
Content-ID
Data
BLOB
MRM
Header
Body
Cookie
SendClaim
Content for each part can be further parsed in a different domain by user

Here is a logical view of a MIME multipart message. The Content-Type values are simple name-value strings. Note that the content for each part can be further parsed in a different domain based on application design.

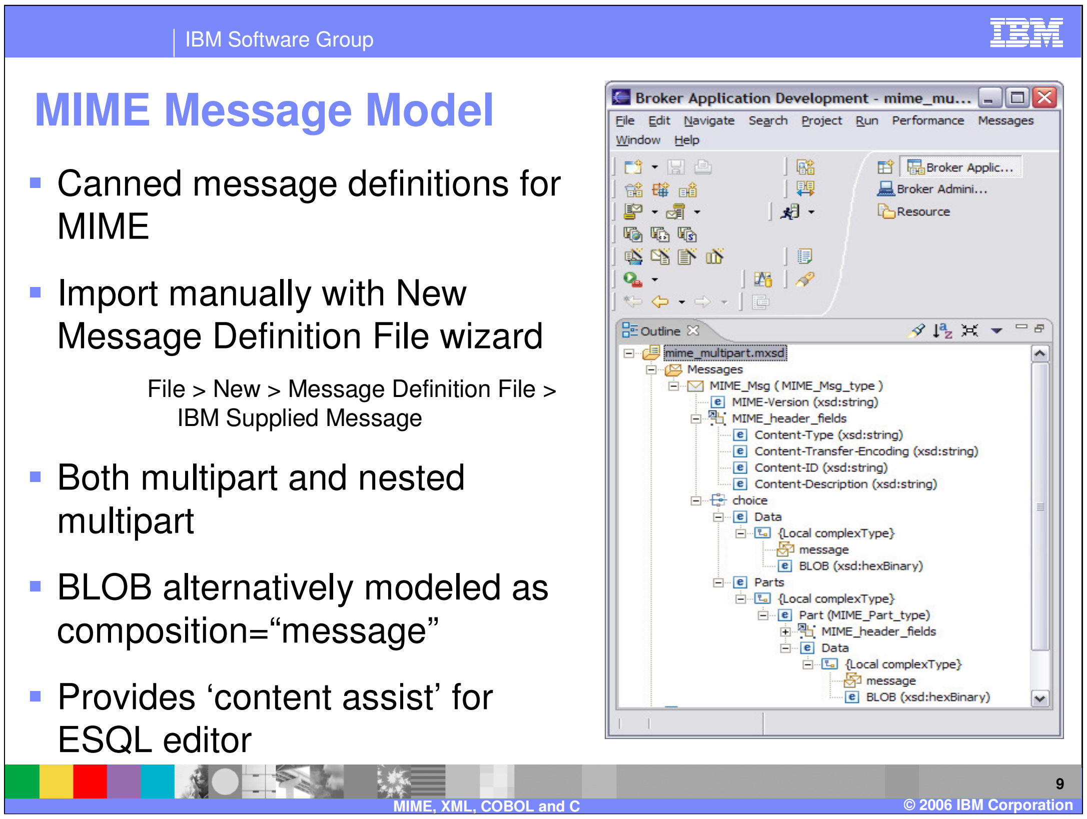
Although Message Broker CAN parse a multipart MIME message, it cannot parse the entire message, including attachments, in one pass. The attachments are blobs after initial MIME parse. For example if your MIME message contains two XML attachments, Message Broker would parse the entire message initially using MIME, then each attachment would be reparsed using the appropriate XML parser. This could be done in a single message flow by propagating each of the attachments to a ResetContentDescriptor node to request that the message is reparsed by a different parser.

MIME – Logical view 2


MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=************; type=text/xml;
Content-Description: An optional message description

****
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: rootpart@example.com
…XML attachment – like SOAP envelope

Here is a logical view of a MIME multipart message, showing header information.

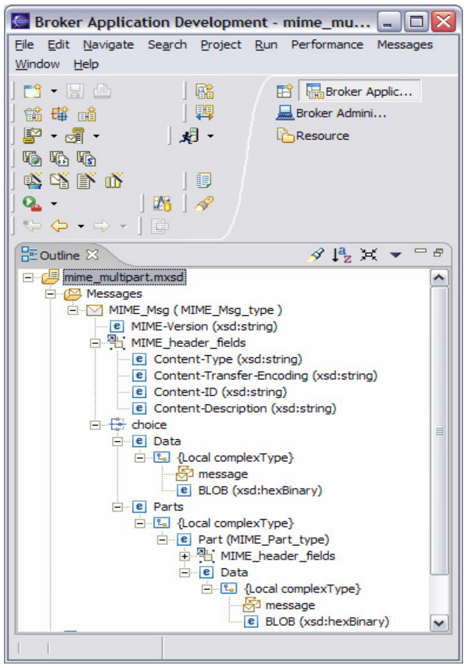


IBM Software Group

MIME Message Model

- Canned message definitions for MIME
- Import manually with New Message Definition File wizard

  File > New > Message Definition File > IBM Supplied Message
- Both multipart and nested multipart
- BLOB alternatively modeled as composition="message"
- Provides 'content assist' for ESQL editor



Broker Application Development - mime_mu...
File Edit Navigate Search Project Run Performance Messages
Window Help
Broker Applic...
Broker Admini...
Resource
Outline

```
mime_multipart.mxsd
  Messages
    MIME_Msg ( MIME_Msg_type )
      MIME-Version (xsd:string)
      MIME_header_fields
        Content-Type (xsd:string)
        Content-Transfer-Encoding (xsd:string)
        Content-ID (xsd:string)
        Content-Description (xsd:string)
      choice
        Data
          {Local complexType}
            message
            BLOB (xsd:hexBinary)
        Parts
          {Local complexType}
            Part (MIME_Part_type)
              MIME_header_fields
              Data
                {Local complexType}
                  message
                  BLOB (xsd:hexBinary)
```

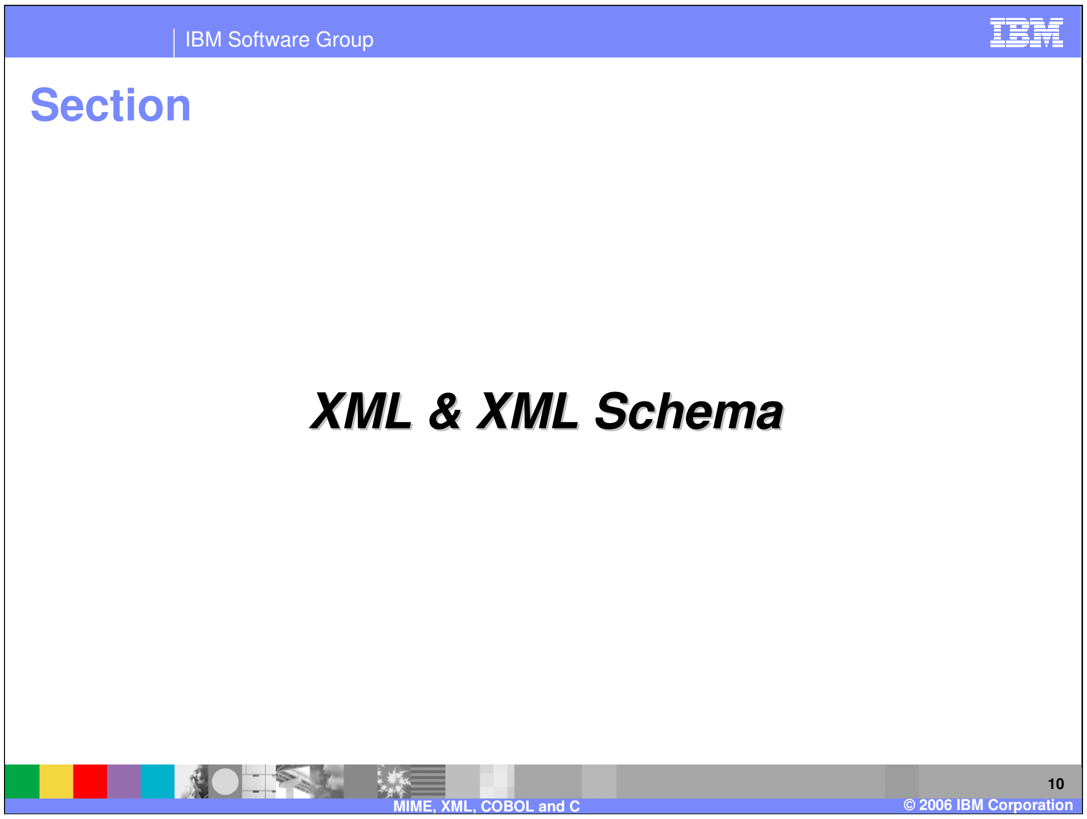
9

MIME, XML, COBOL and C

© 2006 IBM Corporation

To assist in MIME multipart message development, V6 provides canned message definitions for MIME.

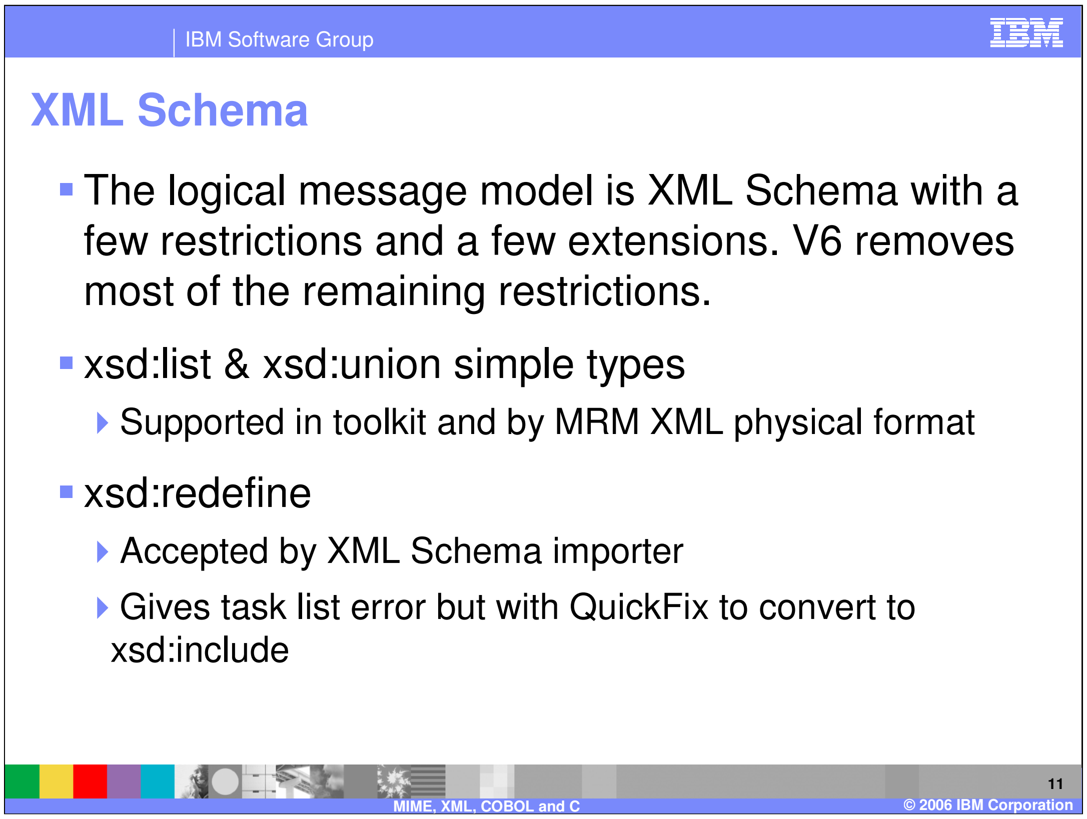
You can import MIME message definitions manually with the new Message Definition File wizard. Both multipart and nested multipart MIME message definitions are supported in the model. The data type of 'BLOB' can alternatively be modeled as composition="message". Within the MIME Message Model, content assist for the ESQL editor is provided.



Section

XML & XML Schema

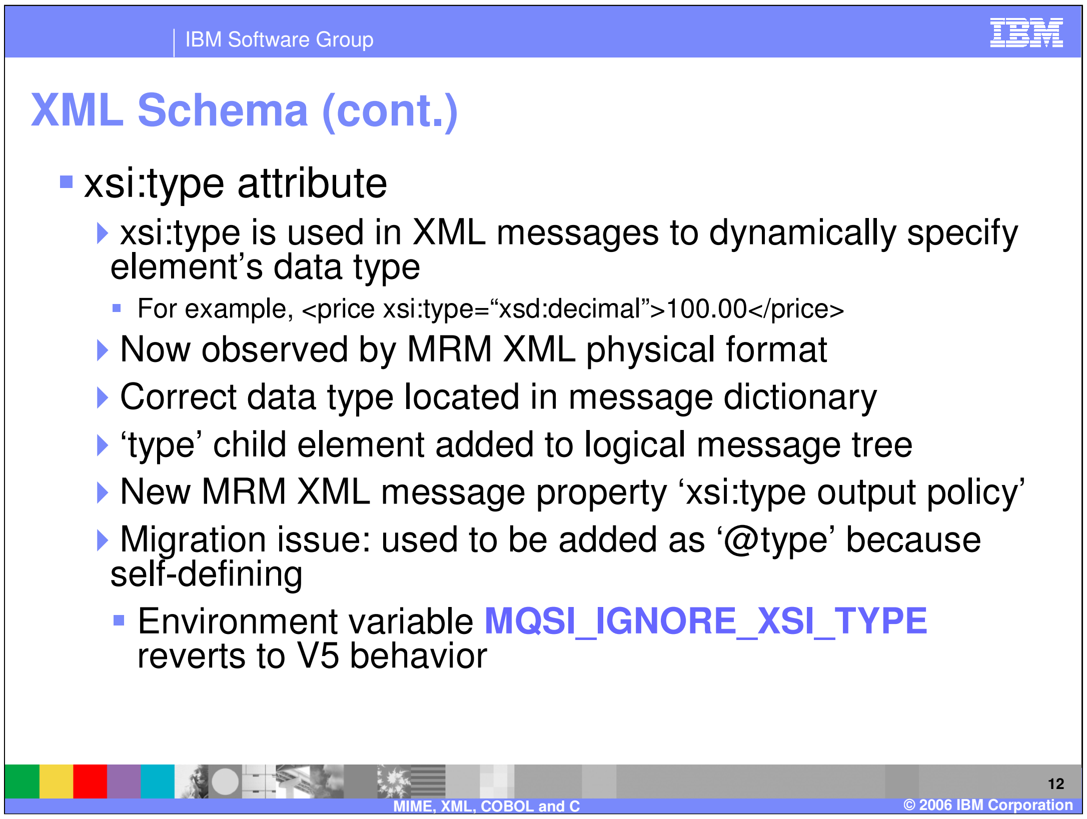
This portion of the presentation discusses the V6 enhancements and changes to XML and XML Schema.



XML Schema

- The logical message model is XML Schema with a few restrictions and a few extensions. V6 removes most of the remaining restrictions.
- xsd:list & xsd:union simple types
  - Supported in toolkit and by MRM XML physical format
- xsd:redefine
  - Accepted by XML Schema importer
  - Gives task list error but with QuickFix to convert to xsd:include

In WebSphere Message Broker, the logical message model is XML Schema. In V6, most of the schema restrictions in earlier releases have been removed. For example, in V6, the xsd:list and xsd:union simple types can be imported without error. In the case of xsd:redefine, it can be imported and accepted by the importer, but a task list error is produced. You can use QuickFix to convert it to xsd:include.

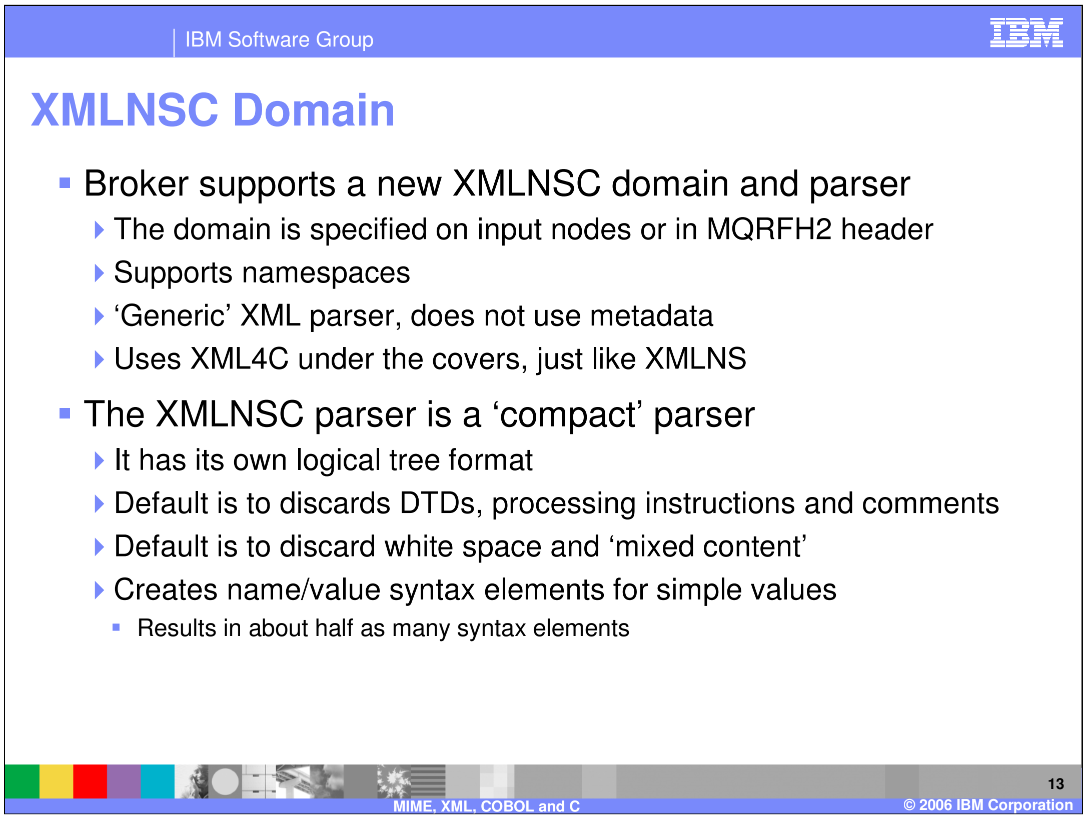


XML Schema (cont.)

- xsi:type attribute
  - xsi:type is used in XML messages to dynamically specify element's data type
    - For example, <price xsi:type="xsd:decimal">100.00</price>
  - Now observed by MRM XML physical format
  - Correct data type located in message dictionary
  - 'type' child element added to logical message tree
  - New MRM XML message property 'xsi:type output policy'
  - Migration issue: used to be added as '@type' because self-defining
    - Environment variable **MQSI_IGNORE_XSI_TYPE** reverts to V5 behavior

In V5, the xsi:type attribute was accepted but support for it was not completely implemented. For example, in V5, when this attribute was detected, the attribute was stored but not fully resolved within the MRM XML physical format. The xsi:type attribute is now fully supported in WebSphere Message Broker V6. The attribute is stored within the physical format and a child element is added to the logical message tree. A new MRM XML message property called 'xsi:type output policy' is provided.

Note there may be a migration issue in V6 because the xsi:type attribute is fully supported; in V5 the attribute was not fully supported and was added as '@type". If you want the V5 behavior, then you can achieve this by setting the environmental variable MQSI_IGNORE_XSI_TYPE.


IBM Software Group

XMLNSC Domain

- Broker supports a new XMLNSC domain and parser
  - The domain is specified on input nodes or in MQRFH2 header
  - Supports namespaces
  - ‘Generic’ XML parser, does not use metadata
  - Uses XML4C under the covers, just like XMLNS
- The XMLNSC parser is a ‘compact’ parser
  - It has its own logical tree format
  - Default is to discards DTDs, processing instructions and comments
  - Default is to discard white space and ‘mixed content’
  - Creates name/value syntax elements for simple values
    - Results in about half as many syntax elements

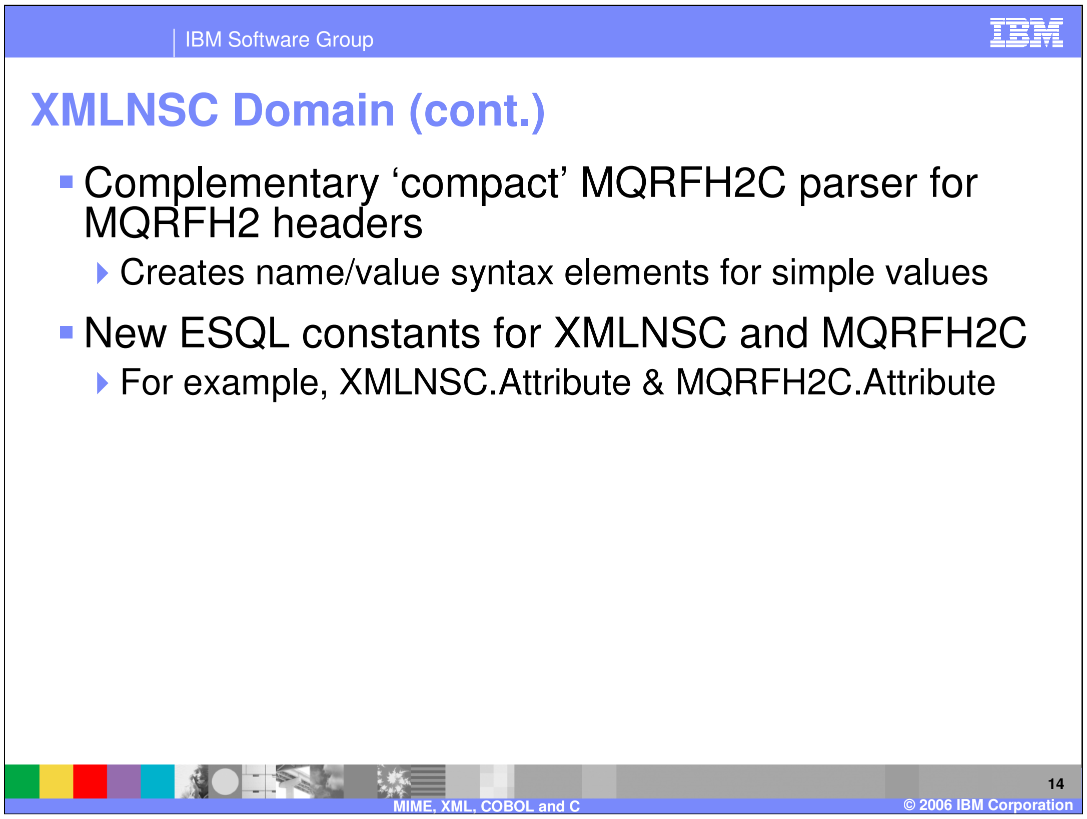
13

MIME, XML, COBOL and C

© 2006 IBM Corporation

In V6, a new way to parse XML messages is provided with the XMLNSC domain and parser. The intent of the XMLNSC parser is to produce “generic” XML that is flexible and very memory-efficient. The XMLNSC parser is an XML parser that is namespace aware, denoted by “NS”, and produces more “compact” formats, denoted by the letter “C”. The domain name is specified on input nodes or in MQRFH2 headers. The XMLNSC parser does not use metadata.

The XMLNSC parser produces more “compact” tree formats and XML than the XMLNS parser in the previous release. The older XMLNS parser would preserve all elements of XML documents during message processing, including white space character formatting, processing instructions and comments, thus increasing the size of the XML document. If the document does not need to preserve white space characters, formatting controls and other such data because the delivery is not a human-readable document, these extra elements in the XML document needlessly increase the document size and processing time. In addition, the older XMLNS parser created individual elements for an element name and an individual element for its value, which also increased XML document size and complexity. By default, the XMLNSC parser discards DTD’s, processing instructions, comments, white space characters, and ‘mixed content’. It also creates a combined name-value syntax element for simple values, resulting in about half as many syntax elements as an equivalent document parsed with XMLNS.



XMLNSC Domain (cont.)

- Complementary ‘compact’ MQRFH2C parser for MQRFH2 headers
  - Creates name/value syntax elements for simple values
- New ESQL constants for XMLNSC and MQRFH2C
  - For example, XMLNSC.Attribute & MQRFH2C.Attribute

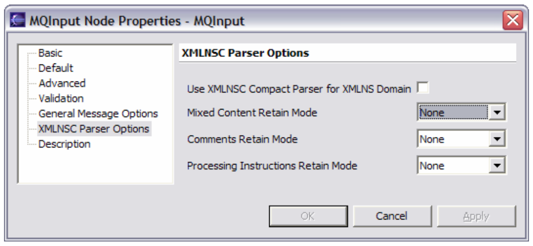
Complementary to the XMLNSC parser “compact” functionality, V6 also provides the MQRFH2C parser for parsing the MQRFH2 headers. The MQRFH2C parser achieves a significant savings in MQRFH2 size and complexity by using the combined “name/value” syntax element for simple values.

Use the MQRFH2C compact parser by selecting the Use MQRFH2C Compact Parser for MQRFH2 Domain check box on the input node of the message flow.

New ESQL constants of XMLNSC.Attribute and MQRFH2C.Attribute settings are provided to indicate compact parsing.

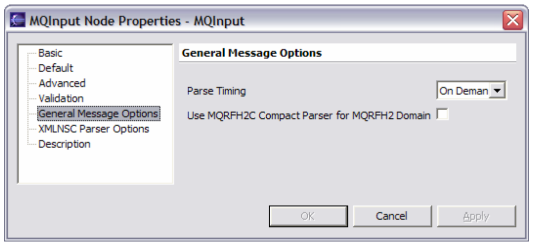
XMLNSC 'Parser Properties'

- New 'Parser Properties' concept
- These are node properties that are passed directly to a parser
- Added for XMLNSC parser to control retention of mixed content, comments and processing instructions



XMLNSC Parser Options

Use XMLNSC Compact Parser for XMLNS Domain ☐
Mixed Content Retain Mode: None
Comments Retain Mode: None
Processing Instructions Retain Mode: None



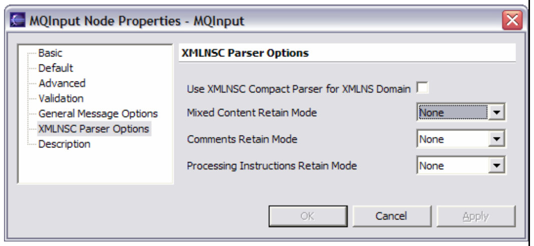
General Message Options

Parse Timing: On Deman
Use MQRFH2C Compact Parser for MQRFH2 Domain ☐

The ability to control the way a parser behaves did not exist until V6. V6 introduces a new 'Parser Properties' concept. These node properties are passed directly to a parser, providing a set of controls to the parser. This new property allows the XMLNSC parser to control the retention of mixed content, comments, and processing instructions.

XMLNSC 'Parser Properties' (cont.)

- Option to invoke XMLNSC parser instead of XMLNS parser
- Option to invoke MQRFH2C parser instead of MQRFH2 parser



MQInput Node Properties - MQInput

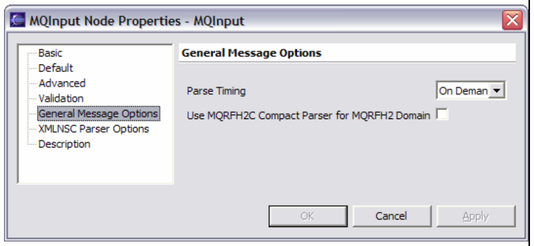
XMLNSC Parser Options

Use XMLNSC Compact Parser for XMLNS Domain ☐

Mixed Content Retain Mode: None

Comments Retain Mode: None

Processing Instructions Retain Mode: None



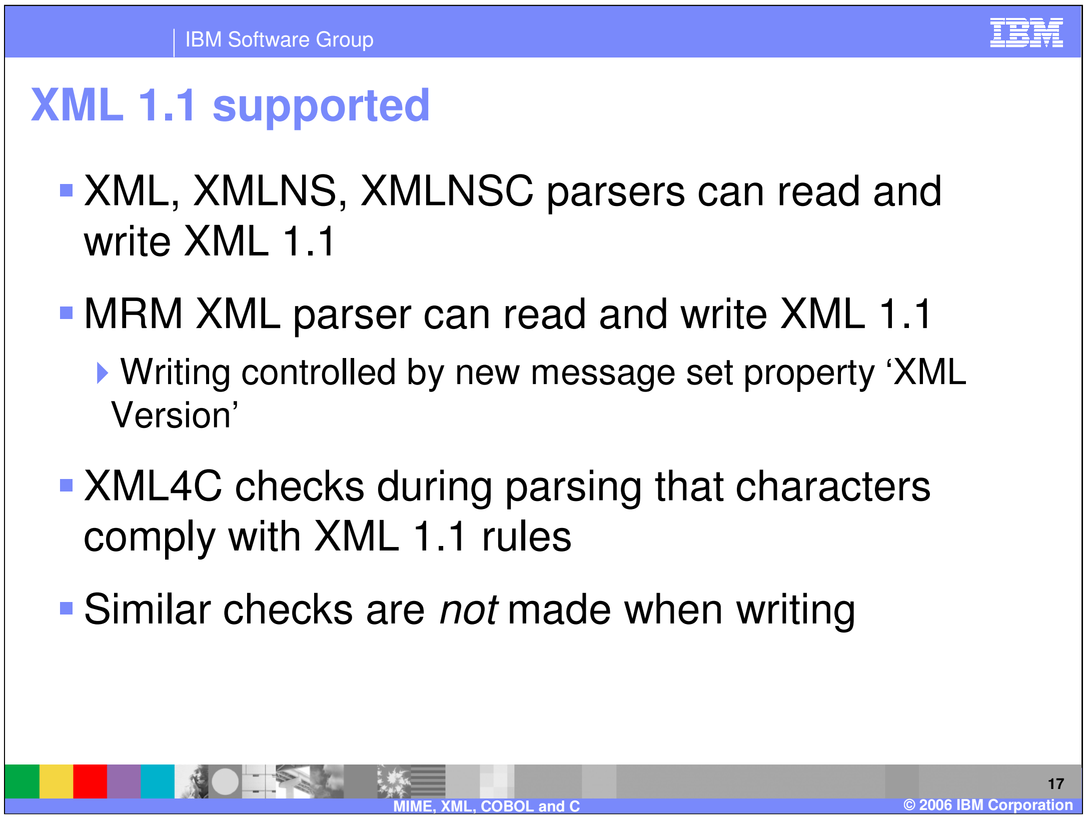
MQInput Node Properties - MQInput

General Message Options

Parse Timing: On Deman

Use MQRFH2C Compact Parser for MQRFH2 Domain ☐

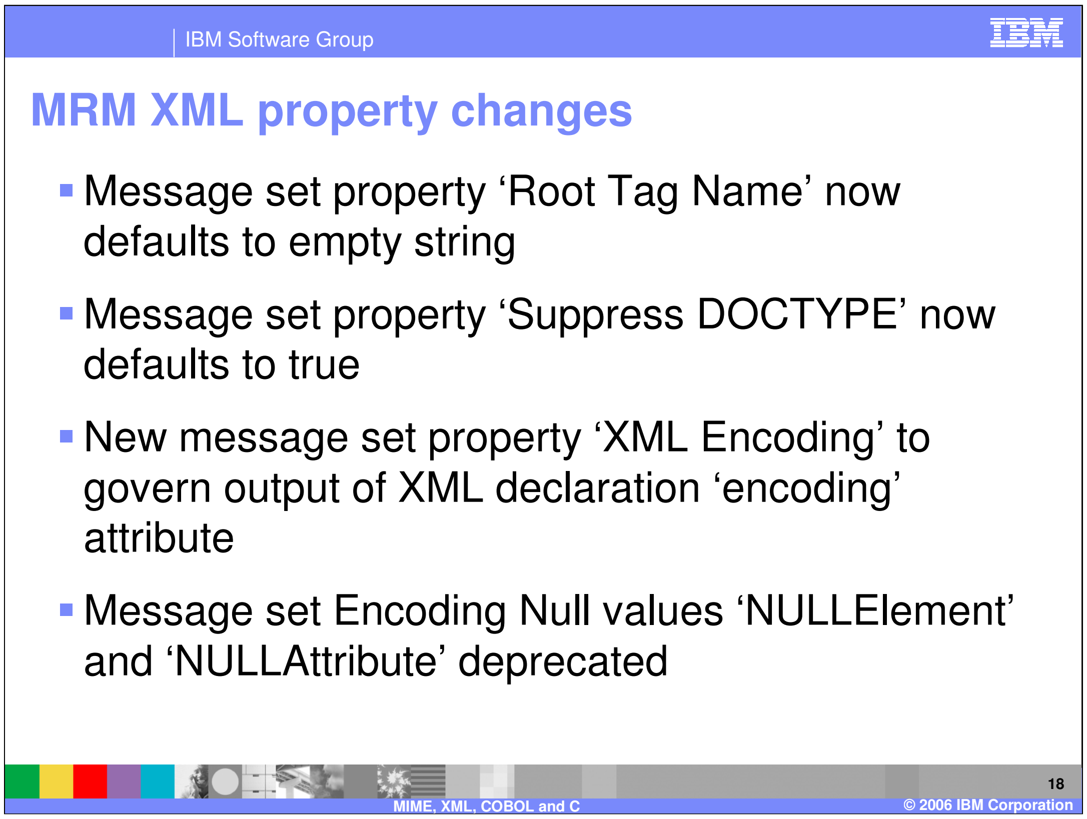
The 'Parser Properties' also sets the option of which parser to invoke. In the graphic, you see that in 'XMLNSC Parser Options' you can set the checkbox 'Use XMLNSC Compact Parser for XMLNS Domain'. In the 'General Message Options', you can set the checkbox for 'Use MQRFH2C Compact Parser for MQRFH2 Domain'.



XML 1.1 supported

- XML, XMLNS, XMLNSC parsers can read and write XML 1.1
- MRM XML parser can read and write XML 1.1
  - Writing controlled by new message set property ‘XML Version’
- XML4C checks during parsing that characters comply with XML 1.1 rules
- Similar checks are *not* made when writing

The XML, XMLNS and XMLNSC parsers can parse and write both XML 1.0 and XML 1.1. In addition, the MRM XML parser can parse and write in either version as well. The version of XML written is controlled by the new message set property ‘XML Version’. The underlying XML4C parser checks during parsing that characters comply with appropriate XML rules. Note that character compliance checks are NOT made when XML documents are written.

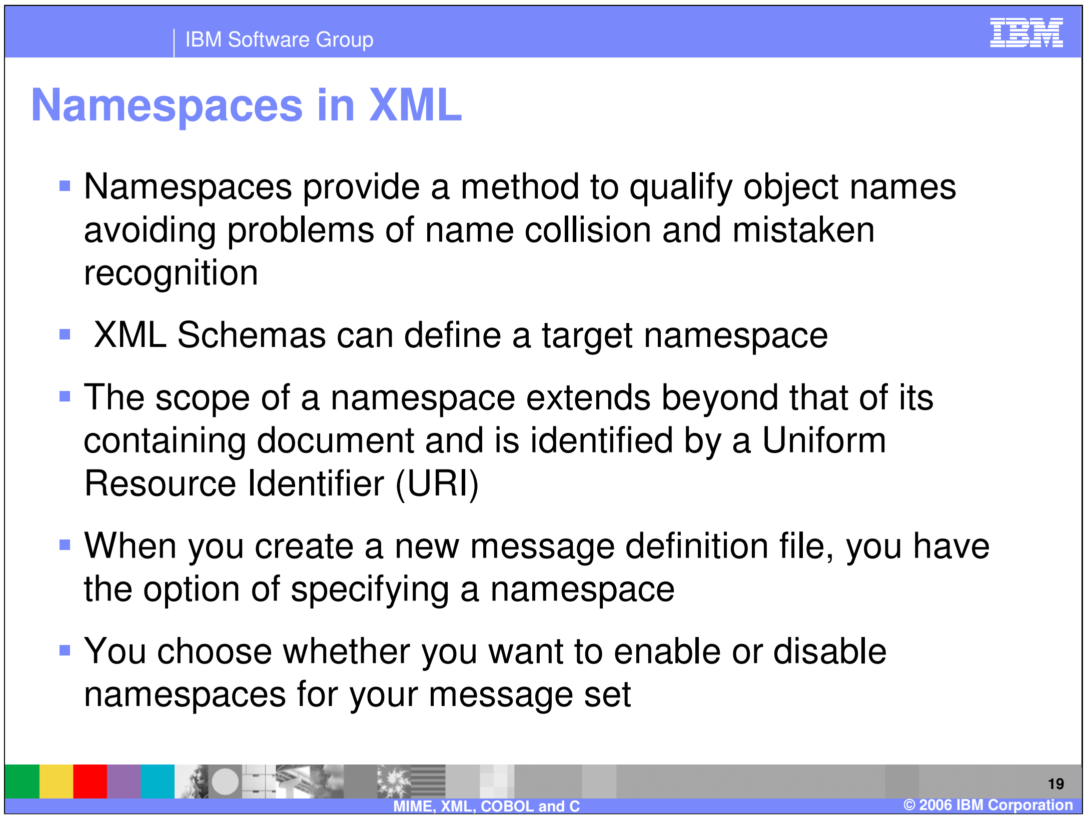


MRM XML property changes

- Message set property ‘Root Tag Name’ now defaults to empty string
- Message set property ‘Suppress DOCTYPE’ now defaults to true
- New message set property ‘XML Encoding’ to govern output of XML declaration ‘encoding’ attribute
- Message set Encoding Null values ‘NULLElement’ and ‘NULLAttribute’ deprecated

Some property changes have been made to reflect more common usage. The message set property ‘Root Tag Name’ now defaults to an empty string. The message set property ‘Suppress DOCTYPE’ now defaults to true.

A new message set property called ‘XML Encoding’ allows you to control the XML declaration ‘encoding’ attribute. The message set Encoding Null values ‘NULLElement’ and ‘NULLAttribute’ are deprecated in V6.


IBM Software Group

Namespaces in XML

- Namespaces provide a method to qualify object names avoiding problems of name collision and mistaken recognition
- XML Schemas can define a target namespace
- The scope of a namespace extends beyond that of its containing document and is identified by a Uniform Resource Identifier (URI)
- When you create a new message definition file, you have the option of specifying a namespace
- You choose whether you want to enable or disable namespaces for your message set

19

MIME, XML, COBOL and C

© 2006 IBM Corporation

XML instance documents and XML Schemas can make use of namespaces. Namespaces provide a method to qualify object names.

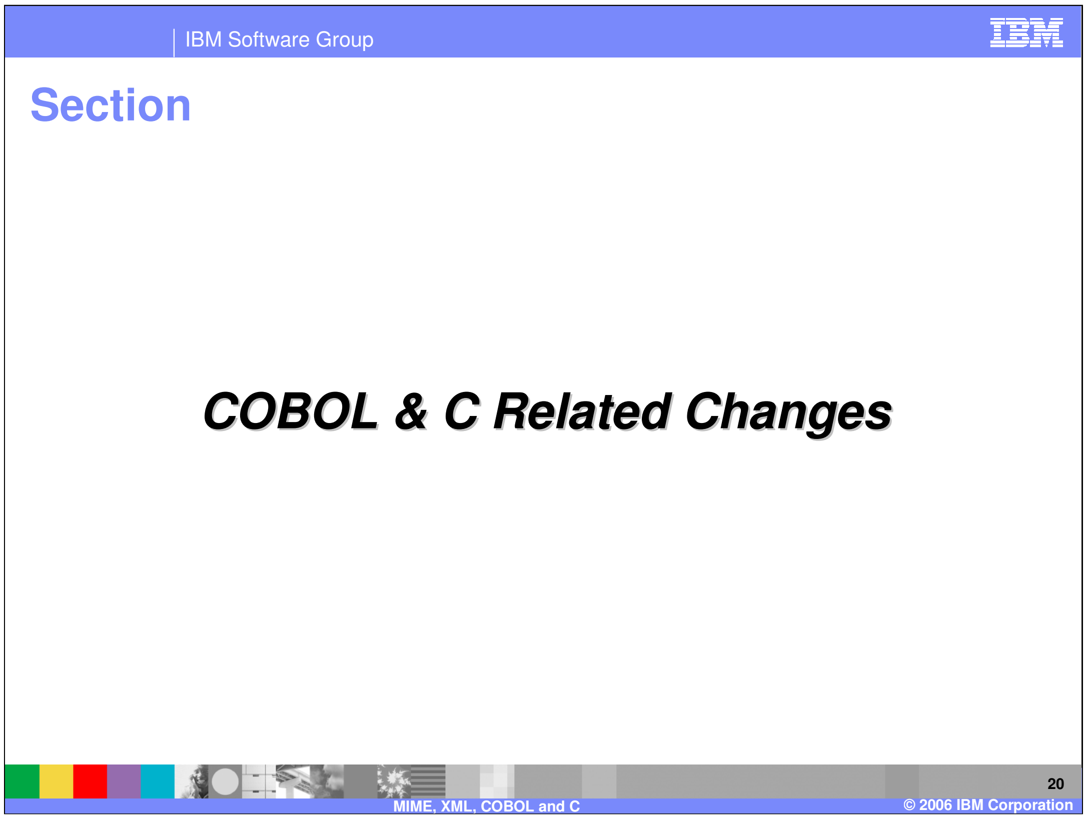
A single XML instance document can contain elements and attributes that are defined for and possibly used by multiple applications. Two different elements or attributes within the same document might require the same name. Individual applications need to be able to recognize the elements and attributes which they are to process. In circumstances such as this, the definitions can be distinguished from each other by qualifying each element with a different namespace. This avoids problems of name collision and mistaken recognition.

XML Schemas can define a target namespace. Global elements, attributes, groups and types defined within an XML Schema are qualified by the target namespace if it has been defined. Optionally, local elements and attributes can also be qualified by the target namespace.

The scope of a namespace extends beyond that of its containing document and is identified by a Uniform Resource Identifier (URI).

A namespace may be associated with a message definition file as it is created.

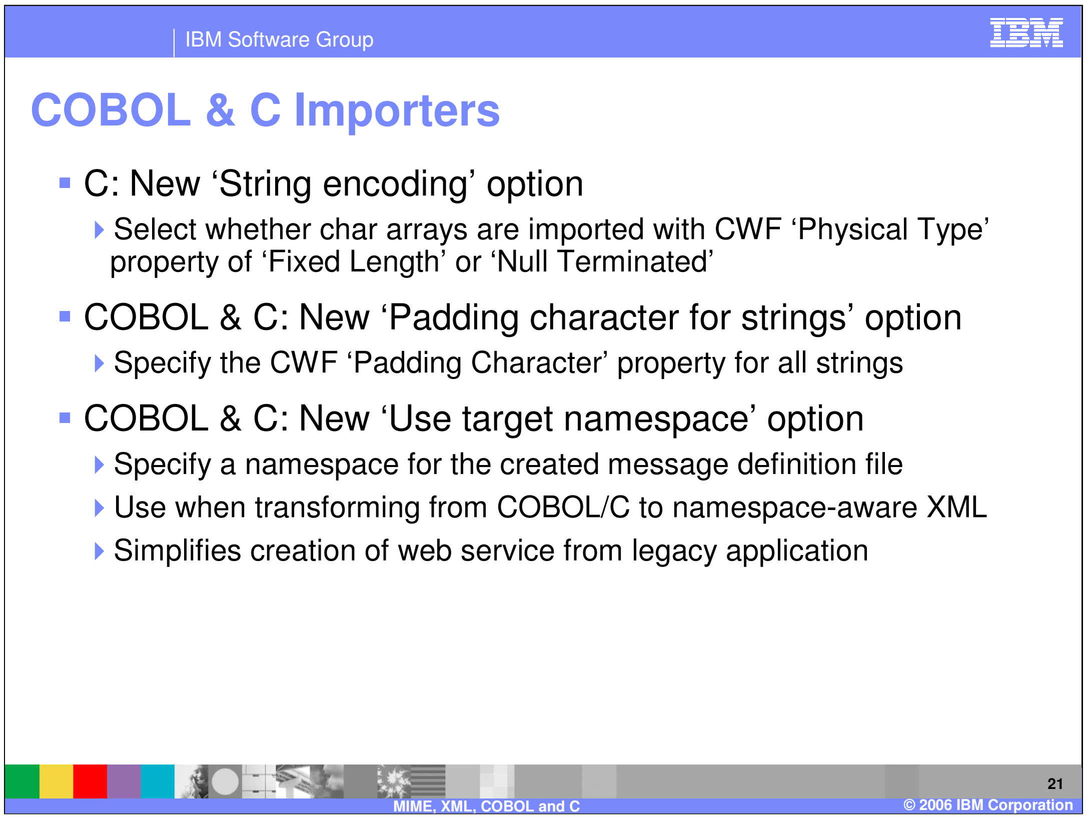
The message model provides the ability to support namespaces within message sets. However you can choose whether you want to enable or disable namespaces for your message set.



Section

COBOL & C Related Changes

This portion of the presentation discusses COBOL and C related changes in V6.

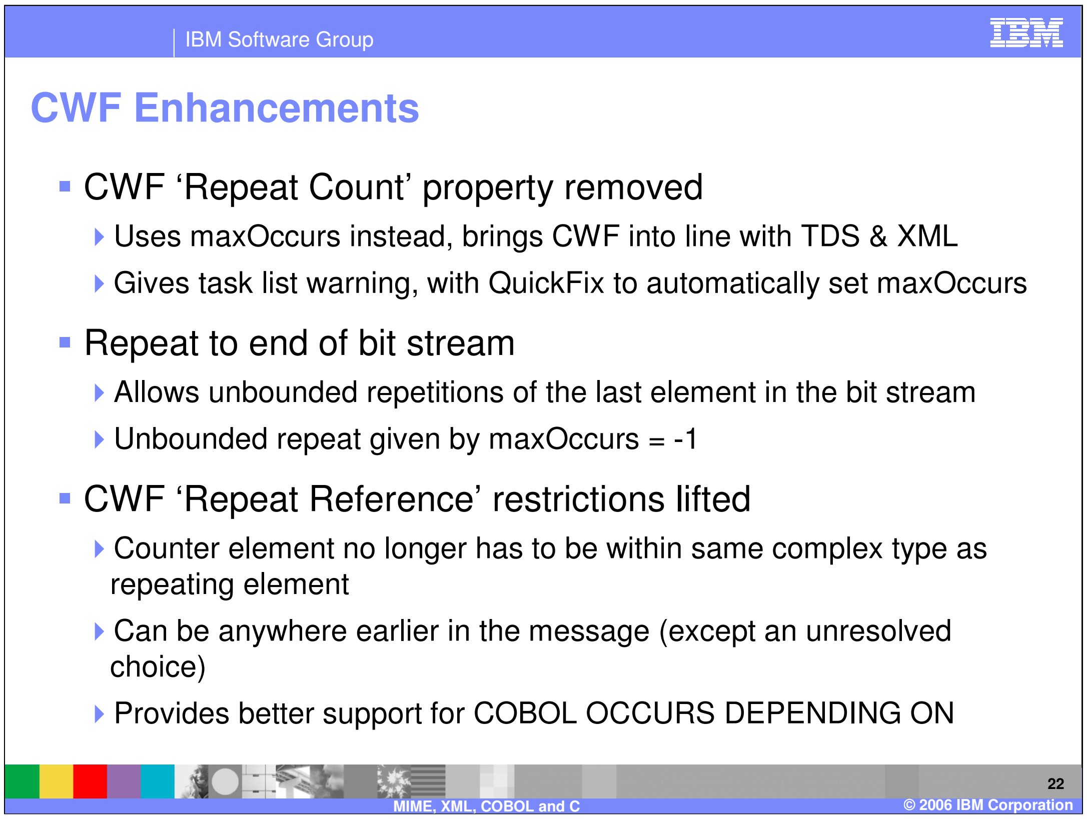


COBOL & C Importers

- C: New ‘String encoding’ option
  - Select whether char arrays are imported with CWF ‘Physical Type’ property of ‘Fixed Length’ or ‘Null Terminated’
- COBOL & C: New ‘Padding character for strings’ option
  - Specify the CWF ‘Padding Character’ property for all strings
- COBOL & C: New ‘Use target namespace’ option
  - Specify a namespace for the created message definition file
  - Use when transforming from COBOL/C to namespace-aware XML
  - Simplifies creation of web service from legacy application

V6 provides a new ‘String encoding’ option for the C importer so you can select whether char arrays are imported with the CWF ‘Physical Type’ property of ‘Fixed Length’ or ‘Null Terminated’.

For the COBOL and C importers, a new ‘Padding character for strings’ option allows you to specify the CWF ‘Padding Character’ for all strings. Also a new ‘Use target namespace’ option allows you to specify a namespace for the created message definition file, used when transforming the COBOL or C input to namespace-aware XML. This feature simplifies the creation of web services from legacy applications.



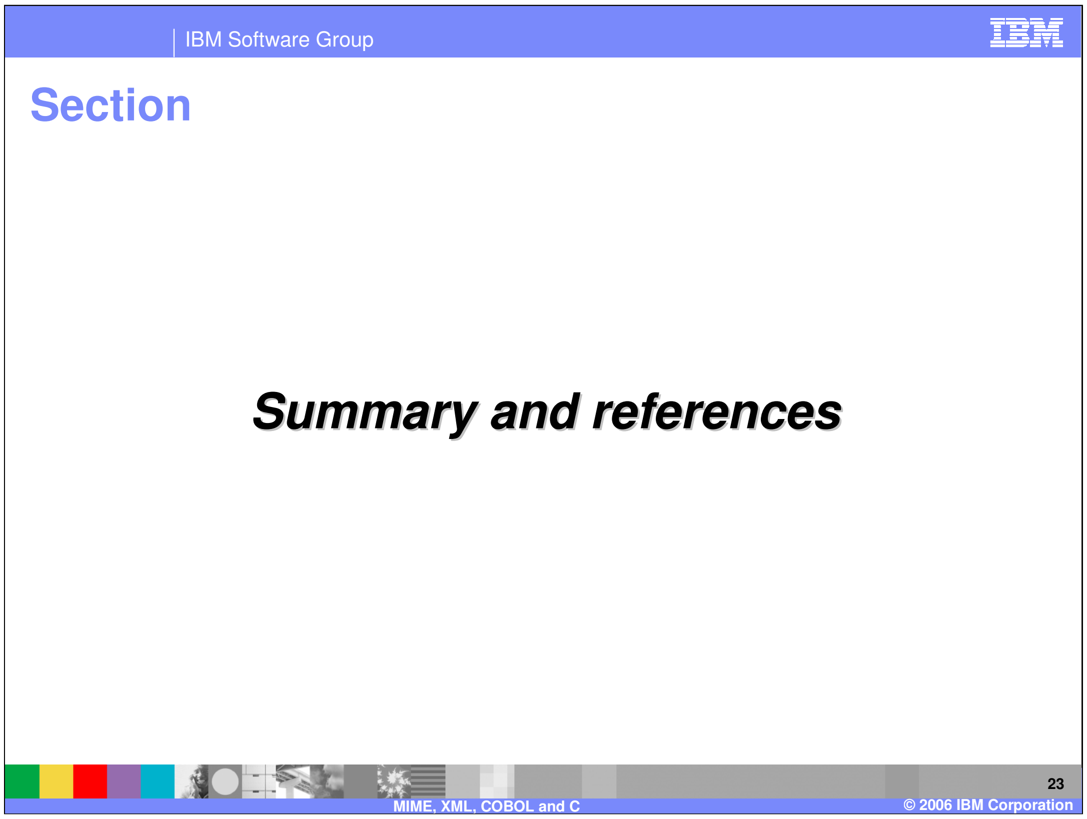
CWF Enhancements

- CWF ‘Repeat Count’ property removed
  - Uses maxOccurs instead, brings CWF into line with TDS & XML
  - Gives task list warning, with QuickFix to automatically set maxOccurs
- Repeat to end of bit stream
  - Allows unbounded repetitions of the last element in the bit stream
  - Unbounded repeat given by maxOccurs = -1
- CWF ‘Repeat Reference’ restrictions lifted
  - Counter element no longer has to be within same complex type as repeating element
  - Can be anywhere earlier in the message (except an unresolved choice)
  - Provides better support for COBOL OCCURS DEPENDING ON

The Custom Wire Format (CWF) ‘Repeat Count’ property is removed. You should instead use ‘maxOccurs’, which is consistent with TDS and XML standards. The CWF ‘Repeat Count’ property will cause a task list warning, with QuickFix available to automatically set this property to ‘maxOccurs’.

A new ‘Repeat to end of bit stream’ property is available, using ‘maxOccurs=-1’. This setting allows unbounded repetitions of the last element in the bit stream until the end of the bit stream.

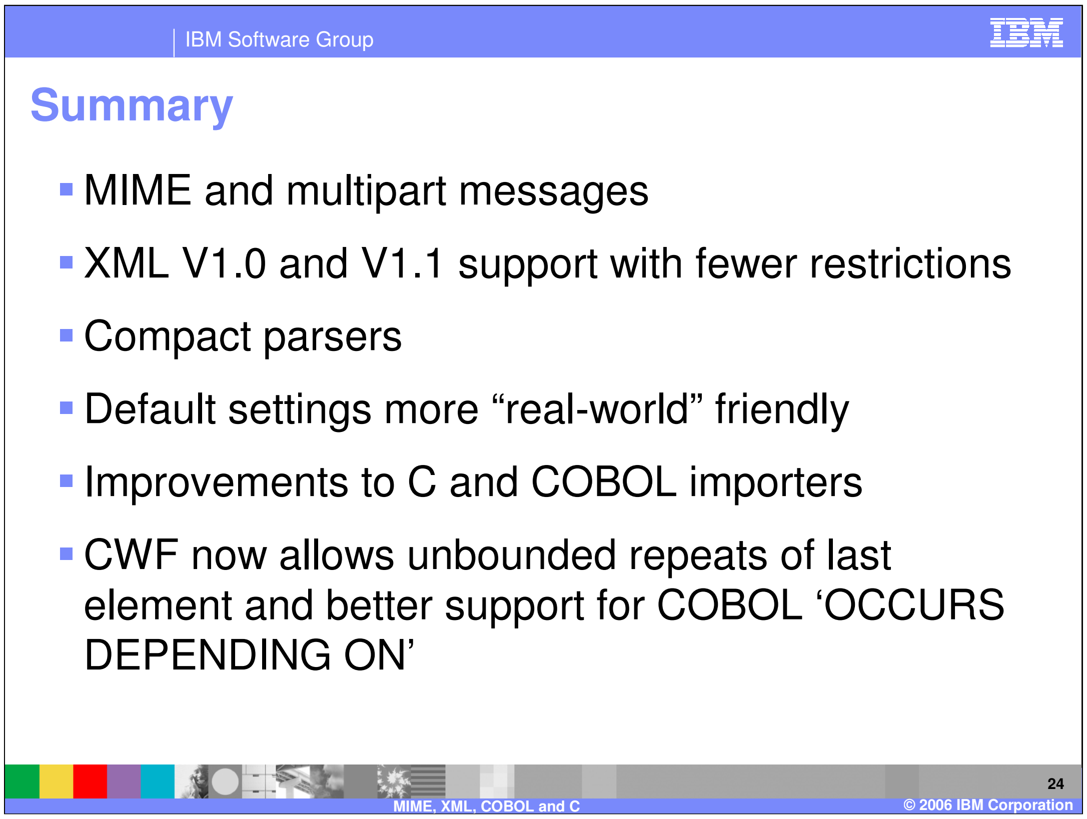
The CWF ‘Repeat Reference’ restrictions have been lifted in V6. The counter element no longer has to be within the same complex type as the repeating element, but instead can be anywhere earlier in the message, with the exception that the counter element cannot be within an unresolved choice. The removal of this restriction provides better support for COBOL ‘OCCURS DEPENDING ON’.



Section

Summary and references

This section contains summary and references.


IBM Software Group

Summary

- MIME and multipart messages
- XML V1.0 and V1.1 support with fewer restrictions
- Compact parsers
- Default settings more “real-world” friendly
- Improvements to C and COBOL importers
- CWF now allows unbounded repeats of last element and better support for COBOL ‘OCCURS DEPENDING ON’

24

MIME, XML, COBOL and C

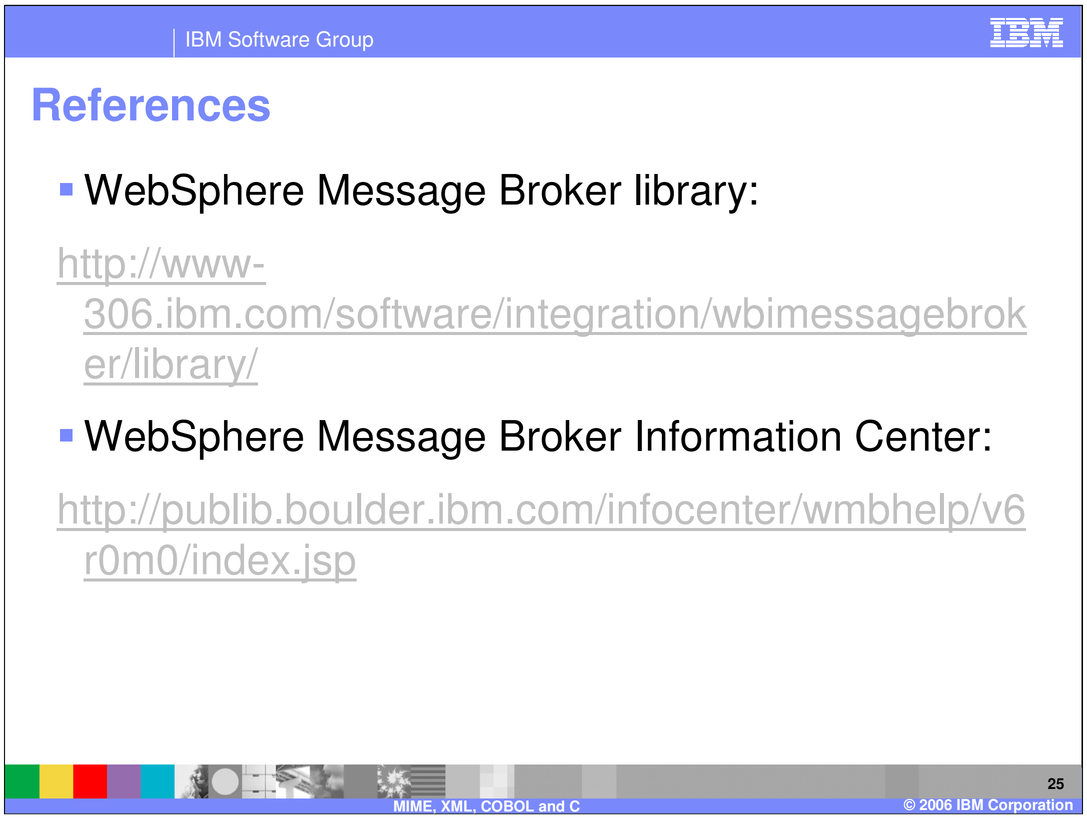
© 2006 IBM Corporation

V6 provides significant enhancements to MIME support with new support for multipart messages.

V6 now supports both XML V1.0 and XML V1.1. Almost all XML element restrictions have been lifted. With the addition of the new compact parsers XMLNSC and XMLRFH2C, tree structures and XML can now be much smaller, saving space and processing time. In addition, unneeded properties have been deprecated and some default settings have been changed to be more consistent with actual usage.

The C and COBOL importers have been improved so that the imported data is more usable, resulting in less time spent in customizing and revising the imported data.

The Custom Wire Format (CWF) has been changed to allow unbounded repeats of the last element. In addition, the CWF offers better support in V6 for COBOL 'OCCURS DEPENDING ON'.



References

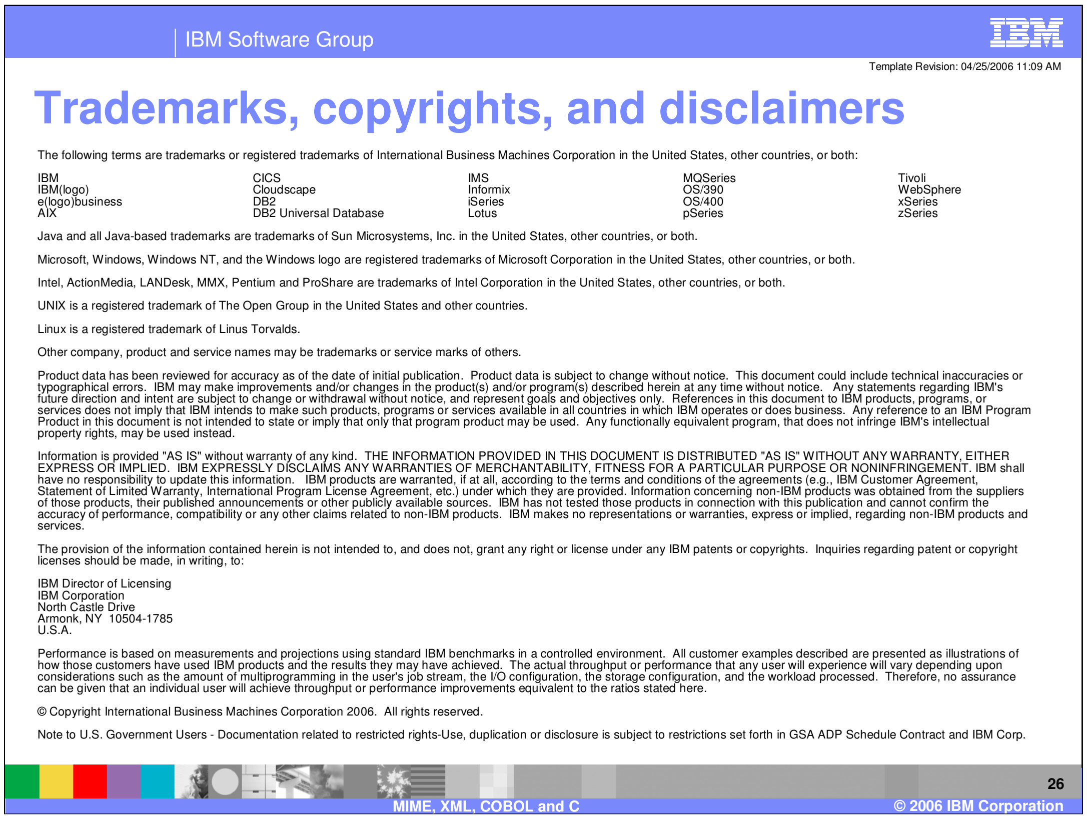
- WebSphere Message Broker library:

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

- WebSphere Message Broker Information Center:

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

References



Template Revision: 04/25/2006 11:09 AM

Trademarks, copyrights, and disclaimers