MQMD/MQRFH2 Parser enhancement

Overview

This enhancement uses the MQMD and MQRFH2 parsers to provide access to the MQMD and MQRFH2 header elements using mapping commands. If the MQ headers are present on the message, the MQMD and MQRFH2 headers are passed to the Message Broker. The Message Broker invokes the MQMD and MQRFH2 parsers as part of the AMM setup, which puts the elements into the AMM. The GetProperty function and SetProperty command are enhanced to allow users to get and set these values. Then the MQMD and MQRFH2 values are recreated using the updated values.

End-User Documentation

Mapping of MQMD and MQRFH2 values

This enhancement allows you to use mapping commands to get and set the values of the MQMD and MQRFH2 headers used by WebSphere MQ. By allowing you to access the values in these MQ headers from your maps, WebSphere Data Interchange can be more easily integrated with other WebSphere MQ and JMS applications.

The header values from the input message can now be read using the "GetProperty" mapping command. The header values on the output message can now be set using the "SetProperty" command.

Getting and setting properties in the MQMD and MQRFH2 headers

To get the values in the MQMD and MQRFH2 headers you use the GetProperty mapping function in your map. To set new values in these headers, you use the SetProperty mapping command.

You must specify the fully qualified path, starting with “ROOT”. Each component of the path is specified by a period (“.”). **Unlike the other properties, this path name is case sensitive**.

The MQMD properties are specified by path “ROOT.MQMD”. For example, “ROOT.MQMD.ReplyToQ” is the ReplyToQ field in the MQMD header.

The MQRFH2 properties are specified by path “ROOT.MQRFH2”. For example “ROOT.MQRFH2.Encoding” is the Encoding field in the MQRFH2 header. Folders within the MQRFH2 folder, such as the “mcd” and “usr” folders can also be specified as part of the path. For example, “ROOT.MQRFH2.mcd.Set” is the “Set” value in the “mcd” folder of the MQRFH2 header.

Sample mapping commands are below:

Get the value of the MQMD MsgId and save it in variable **MyMsgId**:

MyMsgid = GetProperty(“ROOT.MQRFH2.MsgId”)
Get the value of the MQRFH2 Format field and save it in variable Rfh2Fmt:
Rfh2Fmt = GetProperty(“ROOT.MQRFH2.Format”)

Get the value of the domain (msd) from the MQRFH2 *mcd* folder:
MsgDomain = GetProperty(“ROOT.MQRFH2.mcd.Msd”)

Set the value of field “MyField” in the *usr* folder of the MQRFH2 header:
SetProperty(“ROOT.MQRFH2.usr.MyField”, “My user data”)

Some of the fields in the MQMD and MQRFH2 use integer or binary values, instead of the character values used by the GetProperty and SetProperty functions. To allow access to these values, the GetProperty/SetProperty functions will convert from/to character when you get/set MQMD and MQRFH2 fields. When you get these properties from the source message:

- Integer values will be converted to the character representation.
- Binary values will be encoded, similar to the HexEncode function. For example, an 8-byte binary value of x0123456789ABCDEF would be returned as a 16-character string “0123456789ABCDEF”.
- Character values include the blank padding when they are read from fixed-length header fields.

When you set these properties in the target message:

- For integer values, the character string will be converted to an integer.
- For binary values, the encoded character string should be passed, similar to the value passed to HexDecode function. For example, to set an 8-byte binary value of x0123456789ABCDEF you should pass a 16-character string “0123456789ABCDEF”. If the string is too short it will be padded with null characters. If the string is too long, it will be truncated. If unable to decode the string, a warning message will be issued..
- For character values, it will truncate the string or pad with blanks if needed for fixed-length fields.

The supported properties and associated types are listed below:

MQMD properties (ROOT.MQMD.xxx)

Name	Type	Description
StrucId	Char(4)	Structure identifier
Version	Int	Structure version number
Report	Int	Options for report messages
MsgType	Int	Message type
Expiry	Int	Message lifetime
Feedback	Int	Feedback or reason code
Encoding	Int	Numeric encoding of message data
CodedCharSetId	Int	Character set identifier of message data
Format	Char(8)	Format name of message data
Priority	Int	Message priority
Persistence	Int	Message persistence
MsgId	Binary(24)	Message identifier

CorrelId	Binary(24)	Correlation identifier
BackoutCount	Int	Backout counter
ReplyToQ	Char(48)	Name of reply queue
ReplyToQMgr	Char(48)	Name of reply queue manager
UserIdentifier	Char(12)	User identifier
AccountingToken	Binary(32)	Accounting token
ApplIdentityData	Char(32)	Application data relating to identity
PutApplType	Int	Type of application that put the message
PutApplName	Char(28)	Name of application that put the message
PutDate	Char(8)	Date when message was put
PutTime	Char(8)	Time when message was put
ApplOriginData	Char(4)	Application data relating to origin

Following supported only on Windows and AIX (not z/OS and CICS):

GroupId	Binary(24)	Group identifier
MsgSeqNumber	Int	Sequence number of logical message in group
Offset	Int	Offset of data in physical message from start of logical message
MsgFlags	Int	Message flags
OriginalLength	Int	Length of original message

MQRFH2 properties (ROOT.MQRFH2.xxx)

Name	Type	Description
StrucId	Char(4)	Structure identifier
Version	Int	Structure version number
StrucLength	Int	Total length of MQRFH2 including NameValueData
Encoding	Int	Numeric encoding of data that follows NameValueData
CodedCharSetId	Int	Character set identifier of data that follows NameValueData
Format	Char(8)	Format name of data that follows NameValueData
Flags	Int	Flags
NameValueCCSID	Int	Character set identifier of NameValueData

Values in MQRFH2 folders such as the mcd (ROOT.MQRFH2.mcd.xxx) and usr (ROOT.MQRFH2.usr.xxx) are treated as character. No padding or truncation is done.

Other Notes

- The ability to get/set the MQRFH2 values is supported on Windows, AIX, and z/OS. It is not supported on CICS. The ability to get/set the MQMD values is supported on all platforms.
- WDI still sets the following values in the MQRFH2 header as before: Encoding, CodedCharSetId, and the mcd folder values. So if the user sets any of these values, they will get overwritten by WDI-specified values.
- The updated MQMD/MQRFH2 is only used if EDIRFH2 is used as the network program. Network program EDIMQSR continues to use the original MQMD header as before (not the values set in the map), and does not use an MQRFH2 header.

- The MQMD and MQRFH2 headers are not saved in the transaction store. The header values cannot be retrieved or set in the map when doing deferred translation, deferred enveloping, or reenveloping.
- Default values will be used for any MQMD/MQRFH2 values that are not set by the user.
- The MQMD/MQRFH2 values set by the user are not validated by WDI. If the user sets these to invalid values, they may cause errors when the message is written to the queue or when it is received by another application.