Changes to WebSphere MQ message formats during messaging and queuing

Ya Xiao

January 06, 2010

This article describes message format differences between different versions of WebSphere MQ, and discusses changes to messages by WebSphere MQ during the message queuing process. The article is for developers who work with WebSphere MQ and want to understand the evolution of MQ message formats, and changes to messages as they pass through the WebSphere MQ infrastructure.

Introduction

IBM® WebSphere® MQ is IBM’s award-winning middleware for commercial messaging and queuing. Message queuing is a method of program-to-program communication in which applications communicate by sending and retrieving application-specific data in messages through queues, without needing a private, dedicated, logical connection. A message is simply a string of bytes that is meaningful to the applications that use it, and a queue is a container for messages.

This article describes the evolution of WebSphere MQ message formats, and changes to messages made automatically by WebSphere MQ during the message queuing process. You should have some familiarity with developing applications based on WebSphere MQ.

WebSphere MQ message format evolution

A WebSphere MQ message has a typical "header-body" format. In different versions of WebSphere MQ, the header and body vary in content and concept. However, from the first version to WebSphere MQ V6 (hereafter called simply V6), the message format was little changed. In WebSphere MQ V7 (hereafter called V7), to increase compatibility with Java™ Message Service (JMS) applications, message format changed significantly.

Message format through WebSphere MQ V6

Through V6, the two components of WebSphere MQ messages were:

- Message descriptor (Message header)
- Application data (Message body)
Figure 1 shows the message format up through V6:

**Figure 1. Message format through V6**

![Message format diagram](image)

A message descriptor is associated with every message that resides on a queue within a queue manager. It identifies the message and contains control information, such as the type of message and its priority as assigned by the sending application. The application data section contains all actual message data (payload) and any additional headers, such as the transmission header or the IMS information header. WebSphere MQ does not put any restrictions on the content of the application data except a maximum length, which varies from 4 up to 100 MB depending on the version of WebSphere MQ. Maximum message length can also be defined for a queue manager, a channel, or an individual queue.

Messages in different versions up through V6 have the same composition structure, except that:

- The attributes of the message descriptor changed starting with V5 -- see below for details.
- The types of additional headers prefixed to the message data were enriched. For example, the addition of Message descriptor extension (MQMDE) starting with V5, and Embedded PCF header (MQEPH) starting with V6.

1. Message descriptor

The message descriptor is defined by an MQMD structure that contains a number of fields that provide information about the message. The following sections describe two of these attributes in detail.

- **Version**
  
  There are two versions of the message descriptor. Since V5, messages can be segmented or grouped. For this new function, additional fields in message descriptor were needed to keep information for the segmenting and grouping. This enlarged structure is V2, which is the current version of message descriptor. The V2 message descriptor is the same as V1, but has additional fields defined by an MQMDE structure: GroupId, MsgSeqNumber, Offset, MsgFlags, and OriginalLength.

  Other queue managers that don't support these functions (called V1 queue managers) treat the additional information as data. A V2 message descriptor is generally equivalent to using a V1 message descriptor and prefixing the message data with an MQMDE structure. The queue managers that support message segmenting and grouping are referred to as V2 queue managers. Figure 2 shows the relationship between V1 and V2 MQMD:
• Format
Format is a name that the sender of the message uses to indicate to the receiver the nature of the data in a message. The queue manager has a number of built-in formats with names beginning with MQ, such as MQFMT_STRING. Format can also be used to indicate that there is an additional header (extension). For example, if the value of Format field in a message descriptor is MQFMT_CICS, it indicates that the message data begins with the CICS information header MQCIH followed by the payload data. The nature of the payload data is indicated by the Format field of the MQCIH header. Figure 3 shows this use of Format:

Figure 3. Use Format to indicate the nature of data in a message

2. Application data
The application data section contains the actual data sent from one program to another, and its content and structure are defined by the applications that use it. Generally, the application data only contains information that is meaningful to the applications, since the control information in the message descriptor is sufficient. But under some circumstances, more control information is needed and is prefixed to the payload data in the form of additional headers, as shown in Figure 1 above. The additional headers can be added by the applications for a specific use, such as adding an IMS information header (MQIIH) to the application data when sending a message to an IMS bridge. Also, headers are sometimes added by WebSphere MQ, as described below.
Table 1 lists the WebSphere MQ defined headers that can be prefixed to the application data, with their format names.

**Table 1. Additional header structures defined by WebSphere MQ**

<table>
<thead>
<tr>
<th>Header</th>
<th>Header Description</th>
<th>Format Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCIH</td>
<td>CICS information header</td>
<td>MQFMT_CICS</td>
</tr>
<tr>
<td>MQDLH</td>
<td>Dead-letter header</td>
<td>MQFMT_DEAD_LETTER_HEADER</td>
</tr>
<tr>
<td>MQDH</td>
<td>Distribution-list header</td>
<td>MQFMT_DIST_HEADER</td>
</tr>
<tr>
<td>MQEPH</td>
<td>Embedded PCF header</td>
<td>MQFMT_EMBEDDED_PCF</td>
</tr>
<tr>
<td>MQIIH</td>
<td>IMS information header</td>
<td>MQFMT_IMS</td>
</tr>
<tr>
<td>MQMDE</td>
<td>Message-descriptor extension</td>
<td>MQFMT_MD_EXTENSION</td>
</tr>
<tr>
<td>MQCFH</td>
<td>PCF header</td>
<td>MQFMT_ADMIN / MQFMT_EVENT / MQFMT_PCF</td>
</tr>
<tr>
<td>MQRMH</td>
<td>Reference message header</td>
<td>MQFMT_REF_MSG_HEADER</td>
</tr>
<tr>
<td>MQRFH</td>
<td>Formatting header</td>
<td>MQFMT_RF_HEADER</td>
</tr>
<tr>
<td>MQRFH2</td>
<td>Version-2 rules and formatting header</td>
<td>MQFMT_RF_HEADER_2</td>
</tr>
<tr>
<td>MQWIH</td>
<td>Work information header</td>
<td>MQFMT_WORK_INFO_HEADER</td>
</tr>
<tr>
<td>MQXQH</td>
<td>Transmission queue header</td>
<td>MQFMT_XMIT_Q_HEADER</td>
</tr>
</tbody>
</table>

The additional headers can be chained, that is, the application data can optionally contain one or more headers before the payload data. Take the message for publish/subscribe applications before WebSphere MQ V7 as an example -- the message data can start with the MQRFH1 and MQRFH2 headers together. If more than one header is prefixed to the payload data, they are chained in the same way, as shown in Figure 3 above by the Format field.

**Message format in WebSphere MQ V7**

WebSphere MQ V7 (hereafter called V7) introduced the concept of message properties (from JMS), and therefore the two components of a WebSphere MQ message become:

- Message properties (Message header)
- Application data (Message body)

The message format of V7 is shown in Figure 4, as well as the correspondences between the message properties and message descriptor:

**Figure 4. Message format since V7**
1. Message properties

Message properties are a new feature of V7 as part of the Message Queue Interface (MQI). New MQI function calls are used to set, inquire about, and delete message properties (MQSETMP, MQINQMP and MQDLTMP). You can also use message properties to include business data or state information without having to store it in the application data. The use of message properties in WebSphere MQ mimics the use of properties in JMS, which enable you to set properties in a JMS application and retrieve them in a procedural WebSphere MQ application, or vice-versa.

Message properties flow with any message in a WebSphere MQ queue manager, each consisting of a textual name and a value of a particular type. The size of message properties cannot exceed the MaxPropertiesLength setting for a queue manager. Message properties do not count towards the length of the message for the queue and queue manager, but they do count towards the length of the properties as perceived by the queue manager. There is a length limit of 100 MB for message properties, excluding the message descriptor or extension for each message.

2. Message properties represented as MQRFH2

Message properties can be represented as MQRFH2 elements. Figure 5 shows the structure of a MQRFH2 header:

![MQRFH2 header structure](image)

The string placed in NameValueData field consists of a single folder that contains zero or more properties. The content of the folder is treated as message properties if the content=properties element is included in the folder start tag. If every folder in the header contains properties, the MQRFH2 header fields themselves are added to the message properties length and belong to the message properties section. Otherwise, the header fields count towards the message length and are part of application data. Thus, an application can put a message with a buffer larger than the value of MaxMsgLength when the buffer includes properties.

3. Message descriptor fields as properties

Most message descriptor fields, except StructId and Version, can be treated as properties. The property name is constructed by adding a prefix to the message descriptor field's name, as in the example Root.MQMD.Priority. Here Priority is a field in message descriptor. Message descriptor fields are never represented in an MQRFH2 header as for other properties. If the message data
4. Message flowing to a previous version

When a message flows to a queue manager prior to V7, properties of the message, except those in the message descriptor, are treated as message data and count towards the length of the message:

Figure 6. Corresponding relations between messages in V7 and V6

Therefore, the value of the MaxMsgLength attribute of channels going to a system prior to V7 should be raised, to compensate for the fact that more data might be sent for each message.

Changes of messages during message queuing

While the message travels from one program to another through WebSphere MQ, changes may be made to the message to transport it to its destination. This section explores different ways that a message can be changed by WebSphere MQ during transmission. The changes are made automatically by queue managers, message channel agents, or other WebSphere MQ infrastructures, and are transparent to applications.

As you already know, a WebSphere MQ message can be divided into a message header that contains control information, and message data that contains payload data. The payload data is not changed by a queue manager unless data conversion is performed on it. In most cases, changes to messages are made to the control information (headers), such as adding additional headers or modifying existing ones.

Adding additional control information

Under certain circumstances, WebSphere MQ adds extra control information to a message by prefixing additional headers to the message data, and changing the Format field of message descriptor. WebSphere MQ adds header information to messages in the following four situations:

Situation 1. When a message is put on a remote queue
When a message is put on a remote queue, WebSphere MQ adds a transmission header structure (MQXQH) to the message before placing it on the transmission queue. The transmission header contains the name of the destination queue (RemoteQName) and queue manager (RemoteQMgrName), that is, the addressing information. This information is used to route the message to its correct destination in the network. Table 2 lists some main fields in MQXQH:

**Table 2. Main fields in MQXQH**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoteQName</td>
<td>Name of destination queue</td>
</tr>
<tr>
<td>RemoteQMgrName</td>
<td>Name of destination queue manager</td>
</tr>
<tr>
<td>MsgDesc</td>
<td>Original message descriptor (Embedded)</td>
</tr>
</tbody>
</table>

Besides the addressing information, an embedded message descriptor (MsgDesc) is stored within the MQXQH structure as part of the message data. It is the one that came with the original put by the application at the putting end. Another message descriptor is generated by the queue manager when the message is placed on the transmission queue, stored separately from the message data and called a separate message descriptor. Thus, a message on a transmission queue has two message descriptors, as shown in Figure 7:

**Figure 7. Structure of messages on a transmission queue**

The embedded message descriptor is always a V1 MQMD structure. If the message put by the application has non-default values for one or more of the V2 fields in the MQMD, an MQMDE structure follows the MQXQH, and is in turn followed by the application message data, as shown in Figure 7 above.

When the message arrives on its destination queue, the transmission header and the separate message descriptor are stripped off, and the embedded message descriptor then becomes the unique message descriptor of the message -- it is no longer part of the message data.

**Situation 2. When a queue manager cannot deliver a message**

In this case, WebSphere MQ adds a dead-letter header structure (MQDLH) to the message, and puts it on the dead-letter (undelivered-message) queue. Also, the Format field of the message descriptor (MQMD) is changed to indicate that the message contains an MQDLH structure. This
structure includes the name of the destination queue and the reason why the message was put on the dead-letter queue. Table 3 lists the main fields in MQDLH:

### Table 3. Main fields in MQDLH

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>Reason message arrived on dead-letter queue</td>
</tr>
<tr>
<td>DestQName</td>
<td>Name of original destination queue</td>
</tr>
<tr>
<td>DestQMgrName</td>
<td>Name of original destination queue manager</td>
</tr>
<tr>
<td>Format</td>
<td>Format name of data that follows MQDLH</td>
</tr>
<tr>
<td>PutApplType</td>
<td>Type of application that put message on dead-letter queue</td>
</tr>
<tr>
<td>PutApplName</td>
<td>Name of application that put message on dead-letter queue</td>
</tr>
</tbody>
</table>

Messages can be put on the dead-letter queue by queue managers, message channel agents (MCAs), and applications. All messages on the dead-letter queue must be prefixed with an MQDLH, and the messages may be truncated if they are too long for this queue because of the addition of the MQDLH header.

Applications that get messages from the dead-letter queue must verify that the messages begin with an MQDLH structure. The application can determine whether an MQDLH structure is present by examining the Format field in the message descriptor. If the field has the value MQFMT_DEAD_LETTER_HEADER, the message data begins with an MQDLH structure.

### Situation 3. When sending a message to multiple destination queues

In this case, WebSphere MQ adds a distribution header structure (MQDH) to the message, and puts it on the relevant transmission queue. Thus, the application message data is prefixed with MQXQH and MQDH structures. MQDH describes the additional data in a message when that message is a distribution-list message stored on a transmission queue. A distribution-list message is a message sent to multiple destination queues. The additional data consists of the MQDH structure followed by an array of MQOR records and an array of MQPMR records. Table 4 lists the main fields in MQDH:

### Table 4. Main fields in MQDH

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucLength</td>
<td>Length of MQDH structure plus following records</td>
</tr>
<tr>
<td>Format</td>
<td>Format name of data that follows array of MQPMR records</td>
</tr>
<tr>
<td>PutMsgRecFields</td>
<td>Flags indicating which MQPMR fields are present</td>
</tr>
<tr>
<td>RecsPresent</td>
<td>Number of object records present</td>
</tr>
<tr>
<td>ObjectRecOffset</td>
<td>Offset of first object record from start of MQDH</td>
</tr>
<tr>
<td>PutMsgRecOffset</td>
<td>Offset of first put-message record from start of MQDH</td>
</tr>
</tbody>
</table>

The message data of a distribution-list message on a transmission queue has the following sequence:
• MQXQH structure
• MQDH structure plus arrays of MQOR and MQPMR records
• Application message data (payload data)

The structure of a distribution-list message on a transmission queue is shown in Figure 8:

**Figure 8. Structure of distribution-list messages in a transmission queue**

![Diagram showing the structure of a distribution-list message](image)

StrucLength is the number of bytes from the start of the MQDH structure to the start of the message data following the arrays of MQOR and MQPMR records. ObjectRecOffset gives the offset in bytes of the first record in the array of MQOR object records that contain the names of the destination queues. PutMsgRecOffset gives the offset in bytes of the first record in the array of MQPMR put-message records that contain the message properties.

**Situation 4. When the message is a segment or a message in a group**

In this situation, WebSphere MQ may add a message descriptor extension (MQMDE) structure.

Message segmenting can be transparent to the applications. If permitted, the queue manager segments a large message when it does not fit on a queue. If the application is using a V1 MQMD, the queue manager will add an MQMDE structure to the message data automatically. As already discussed above, the MQMDE structure contains those MQMD fields that exist in the V2 MQMD, but not in the V1 MQMD, where the segmenting and grouping information are stored.

**Changes in the fields of existing headers**

Besides adding extra headers to a message, WebSphere MQ might also alter some specific fields of existing headers in a message during its travels between the sending and receiving applications:

1. **Format field in all preceding headers**

As mentioned above, WebSphere MQ adds header information to the message in specific circumstances. Once an extra header is added, the Format field of its preceding header structure is also updated by WebSphere MQ to indicate the type of the new header. For example, when WebSphere MQ adds a transmission header following the message descriptor of a message, the Format field of MQMD is changed to MQFMT_XMIT_Q_HEADER.
2. Fields containing addressing information in transmission header

During intercommunication (in WebSphere MQ, intercommunication means sending messages from one queue manager to another), when a queue manager passes a message through, it may alter the addressing information (RemoteQName, RemoteQMgrName) of the transmission header transmitted together with the message, if a queue manager alias is used.

Messages arriving from the adjacent system contain the physical name (after name resolution) of the destination queue manager and the queue in the transmission header. If a queue manager alias definition exists with the same queue manager name as that referenced in the transmission header, then the local queue manager, through which the message is passed, overwrites the RemoteQMgrName field with the RQMNAME from that alias definition.

3. ReplyToQ, ReplyToQMgr in message descriptor

Actually, the changes of these two fields are made before a message is put on a queue by applications. If an application puts a message to a queue providing the name of a reply-to queue (ReplyToQ) for answer messages but with the queue manager name (ReplyToQMgr) blank, the local queue manager responds to the blank queue manager name by checking for a remote queue definition with the same name as the reply-to queue (reply-to queue alias):

- If none is found, the queue manager places its own name in the ReplyToQMgr field of the message descriptor, with ReplyToQ unchanged.
- If the definition exists, the queue manager extracts the queue name and queue manager names from the definition and puts them into the reply-to fields of the message descriptor -- that is, it replaces the values of ReplyToQ and ReplyToQMgr.

4. Context fields in message descriptor

The message context information is stored in the context fields of the message descriptor. There are two types of message context: identity context and origin context. A new type of context information is added in V7 for message properties: user context. Table 5 lists the context fields in the message descriptor:

<table>
<thead>
<tr>
<th>Type of Context</th>
<th>Fields and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity context</td>
<td><strong>UserIdentity</strong>: User identifier of the application that originated the message.</td>
</tr>
<tr>
<td></td>
<td><strong>AccountingToken</strong>: Accounting token.</td>
</tr>
<tr>
<td></td>
<td><strong>ApplIdentityData</strong>: Application data relating to identity.</td>
</tr>
<tr>
<td>Origin context</td>
<td><strong>PutAppType</strong>: Type of application that put the message.</td>
</tr>
<tr>
<td></td>
<td><strong>PutAppName</strong>: Name of the application that put the message.</td>
</tr>
<tr>
<td></td>
<td><strong>PutDate</strong>: Date when message was put.</td>
</tr>
<tr>
<td></td>
<td><strong>PutTime</strong>: Time when message was put.</td>
</tr>
</tbody>
</table>
Most fields of the identity and origin context are usually supplied by the queue manager. Of course, applications with appropriate authority can provide their own context:

- Identity context information identifies the user of the application that first put the message on a queue. The queue manager fills the UserIdentifier and AccountingToken fields based on the environment in which the application is running.
- Origin context information describes the application that put the message on the queue on which the message is currently stored. The following fields are usually specified by the queue manager: PutApplType, PutApplName, PutDate, PutTime.
- User context is not set by the queue manager automatically.

5. Segmenting/Grouping fields in message descriptor (V2)

If allowed, the queue manager segments a message into a number of smaller messages when it is too large to be sent. In the meantime, the queue manager sets the following segmenting/grouping fields in the message descriptor if it is V2: GroupId, MsgSeqNumber, Offset, MsgFlags, OriginalLength. As discussed above, an MQMDE structure is added by the queue manager if the message descriptor is V1.

Changes caused by data conversion

If the data formats between the sending and receiving applications differ, data conversion is necessary. Because WebSphere MQ must be able to understand the contents of the message descriptor (MQMD) regardless of the platform it is created on, the MQMD is converted automatically by the system. However, for the application data section of the message, it is a little complicated:

- **Data in WebSphere MQ defined formats (built-in formats)** -- The formats listed in Table 1 are a subset of all built-in formats. Message data in these formats can be converted automatically by a queue manager from one coded character set to another, provided that both character sets are related to a single language or a group of similar languages.
- **Data in user-defined formats** -- The queue manager cannot convert data in user-defined formats. Applications have to supply a data-conversion exit for such formats.

In summary, only the message data in built-in formats can be converted automatically by WebSphere MQ when data conversion is necessary.

Conclusion

This article described the WebSphere MQ message format and message changes made by WebSphere MQ during message queuing. It showed how WebSphere MQ message formats in different versions are very different in concept, but similar in composition, with a header-body structure with two components:
• Message descriptor and application data (in versions up through V6).
• Message properties and application data (in V7).

A message may be changed by WebSphere MQ while it travels from one application to another in three ways:

• Extra headers (containing more control information) may added.
• Fields of the existing headers in the message may be altered.
• Data conversion may be performed.
Related topics

- **WebSphere MQ developer resources page**
  Technical resources to help you design, develop, and deploy messaging middleware with WebSphere MQ to integrate applications, Web services, and transactions on almost any platform.

- **WebSphere MQ product page**
  Product descriptions, product news, training information, support information, and more.

- **WebSphere MQ V6 information center**
  A single Web portal to all WebSphere MQ V6 documentation, with conceptual, task, and reference information on installing, configuring, and using WebSphere MQ V6.

- **IBM Redbook: WebSphere MQ V6 fundamentals**
  Describes the fundamental concepts and benefits of message queuing technology, and provides a technical description of the WebSphere MQ product.

- **WebSphere MQ V7 information center**
  A single Web portal to all WebSphere MQ V7 documentation, with conceptual, task, and reference information on installing, configuring, and using WebSphere MQ V7.

- **IBM Redbook: WebSphere MQ V7 features and enhancements**
  Describes the fundamental concepts and benefits of message queuing technology, describes the new features in V7, and provides a business scenario that shows those features in action.

- **WebSphere MQ V7 trial download**
  A 90 day, full featured no-charge trial of WebSphere MQ V7.0

- **WebSphere MQ documentation library**
  WebSphere MQ product manuals.

- **WebSphere MQ support page**
  A searchable database of support problems and their solutions, plus downloads, fixes, problem tracking, and more.

- **WebSphere MQ public newsgroup**
  A non-IBM forum where you can get answers to your WebSphere MQ technical questions and share your WebSphere MQ knowledge with other users.

- **WebSphere MQ SupportPacs**
  Downloadable code, documentation, and performance reports for the WebSphere MQ family of products.

- **developerWorks WebSphere application connectivity developer resources**
  How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application connectivity and business integration solutions.

- **developerWorks WebSphere business process management developer resources**
  WebSphere BPM how-to articles, downloads, tutorials, education, product info, and other resources to help you model, assemble, deploy, and manage business processes.

- **Most popular WebSphere trial downloads**
  No-charge trial downloads for key WebSphere products.

- **WebSphere on-demand demos**
  Download, watch, and learn what WebSphere products and WebSphere-related technologies can do for your company.

- **developerWorks WebSphere weekly newsletter**
The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, Web services, and other topics. Subscribe now and design your custom mailing.

- **WebSphere-related books from IBM Press**
  Convenient online ordering through Barnes & Noble.
- **developerWorks blogs**
  Join a conversation with developerWorks users and authors, and IBM editors and developers.
- **developerWorks on Twitter**
  Check out recent Twitter messages and URLs.

© Copyright IBM Corporation 2010
**Trademarks**
(www.ibm.com/developerworks/ibm/trademarks/)