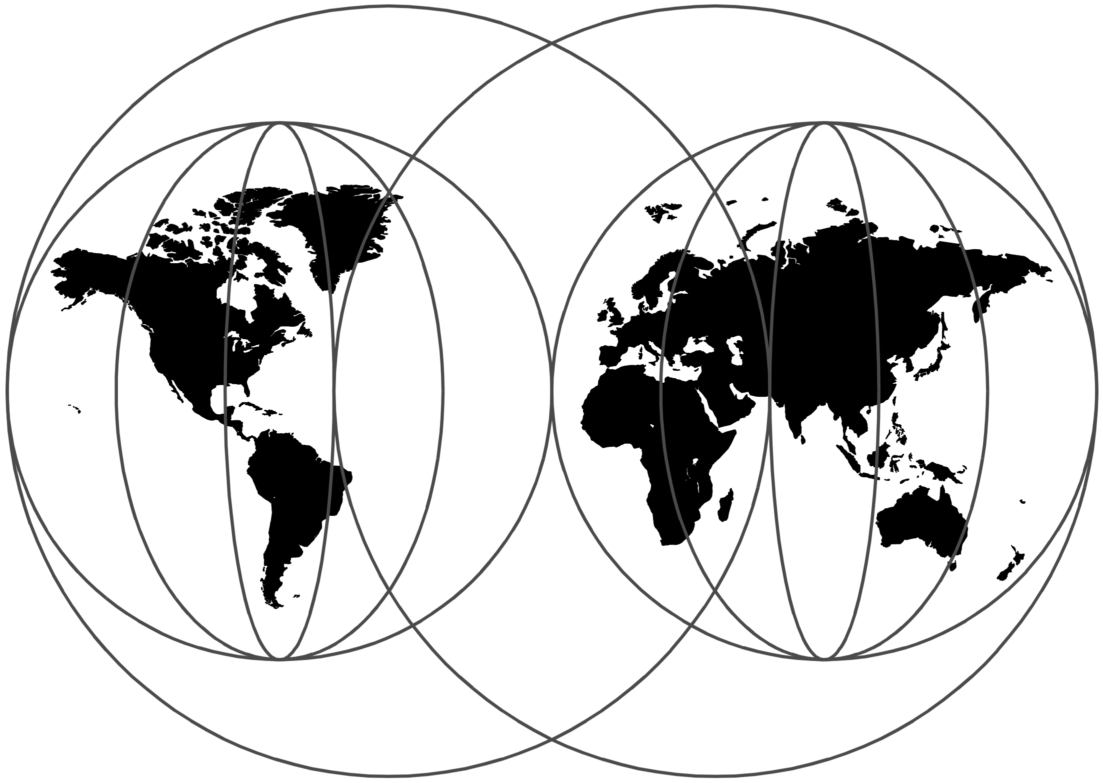
MQSeries Primer



MQSeries Enterprise Application Integration Center

Dieter Wackerow

Find authenticated court documents without watermarks at docketalarm.com.

Find authenticated court documents without watermarks at docketalarm.com.

MQSeries is IBM's award winning middleware for commercial messaging and queuing. It is used by thousands of customers in every major industry in many countries around the world. MQSeries speeds implementation of distributed applications by simplifying application development and test.

MQSeries runs on a variety of platforms. The MQSeries products enable programs to communicate with each other across a network of unlike components, such as processors, subsystems, operating systems and communication protocols. MQSeries programs use a consistent application program interface (API) across all platforms.

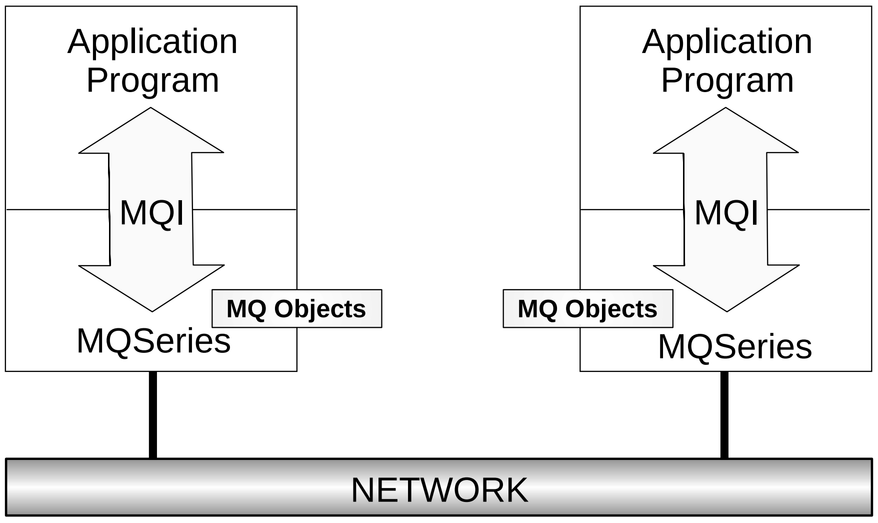


Figure 1. MQSeries at Run Time

Figure 1 shows the main parts of an MQSeries application at run time. Programs use MQSeries API calls, that is the Message Queue Interface (MQI), to communicate with a queue manager (MQM), the run-time program of MQSeries. For the queue manager to do its work, it refers to objects, such as queues and channels. The queue manager itself is an object as well.

The following provides a brief overview of MQSeries, including clients and servers.

What is Messaging and Queuing?

Message queuing is a method of program-to-program communication. Programs within an application communicate by writing and retrieving application-specific data (messages) to/from queues, without having a private, dedicated, logical connection to link them.

Messaging means that programs communicate with each other by sending data in messages and not by calling each other directly.

Queuing means that programs communicate through queues. Programs communicating through queues need not be executed concurrently.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

With *asynchronous messaging*, the sending program proceeds with its own processing without waiting for a reply to its message. In contrast, *synchronous messaging* waits for the reply before it resumes processing. For the user, the underlying protocol is transparent. The user is concerned with conversational or data-entry type applications.

MQSeries is used in a client/server or distributed environment. Programs belonging to an application can run in one workstation or in different machines on different platforms. Applications can easily be moved from one system or platform to another. The programs can be written in various programming languages, including Java. The same queuing mechanism is valid for all platforms, and so are the currently 13 APIs.

Since MQSeries communicates via queues it can be referred to as using indirect program-to-program communication. The programmer cannot specify the name of the target application to which a message is sent. However, he or she can specify a target queue name; and each queue is associated with a program. An application can have one or more "input" queues and may have several "output" queues containing information for other servers to be processed, or for responses for the client that initiated the transaction.

The programmer does not have to worry about the target program being busy or not available. He or she isn't even concerned about the server being down or having no connection to it. The programmer sends messages to a queue that is associated with an application; and the application may or may not be available at the time of the request. MQSeries takes care of the transport to the target application and even starts it, if necessary.

If the target program is not available, the messages stay in a queue and get processed later. The queue is either in the sending machine or in the target machine, depending whether the connection between the two systems can be established or not. Applications can be running all day long or they can be triggered, that is, automatically started when a message arrives or after a specified number of messages have arrived.

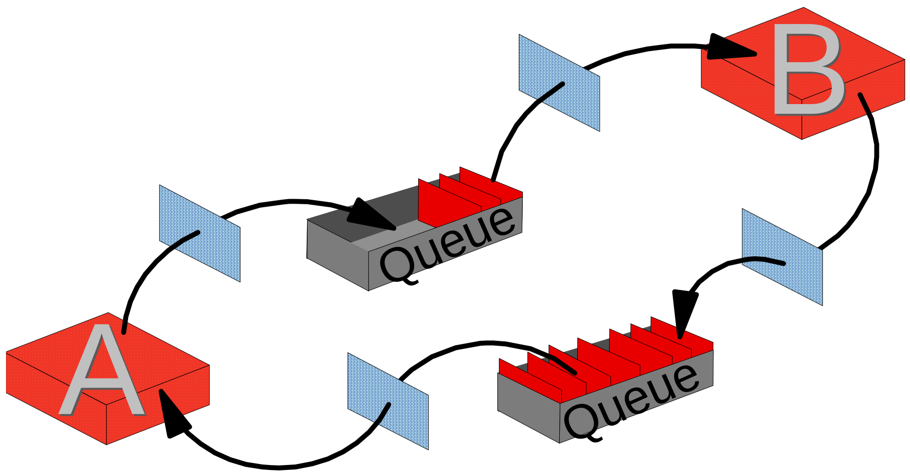


Figure 2. Messages and Queues

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

Figure 2 on page 4 shows how two programs, A and B, communicate with each other. We see two queues; one is the “output” queue for A and at the same time the “input” queue for B, while the second queue is used for replies flowing from B to A.

The squares between the queues and the programs represent the Message Queuing Interface (API) the program uses to communicate with MQSeries’ run-time program, the queue manager. As said before, the API is a simple multi platform API consisting of 13 calls. The API will be discussed later.

About Messages

A message consists of two parts:

1. Data that is sent from one program to another
2. The message descriptor or message header

The message descriptor identifies the message (message ID) and contains control information, also called attributes, such as message type, expiry time, correlation ID, priority, and the name of the queue for the reply.

A message can be up to 4 MB or 100 MB long, depending on the MQSeries version you use. MQSeries Version 5 (for distributed platforms) supports a maximum message length of 100 MB.

Message Segmenting and Grouping

In MQSeries Version 5, messages can be *segmented* or *grouped*. Message segmenting can be transparent to the application programmer. If permitted, the queue manager segments a large message when it does not fit in a queue. On the receiving end, the application has the option to either receive the entire message in one piece or each segment separately. This may depend on the buffer size available for the application.

A second method of segmenting leaves the programmer in control so that he or she can split a message according to logical boundaries or buffer size available for the program. The programmer puts each segment as a separate physical message; thus several physical messages build one logical message. The queue manager ensures that the order of the segments is maintained.

To reduce traffic over the network, you can also group several small messages together and build one larger physical message. This message is then sent to the destination and is there disassembled. Message grouping also guarantees that the order the messages are sent in is preserved.

Distribution Lists

Using MQSeries Version 5, you can send a message to more than one destination queue with one MQPUT call. This is done with a dynamic *distribution list.* A distribution list can be a file that is read at the time an application starts. It can be modified any time. It contains a list of queue names and the queue managers that own them. A message sent to multiple queues belonging to the same queue manager is sent over the network only once and so reduces network traffic. The

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.