VisualAge Generator V4.5 Simplifies Developing MQSeries Business Integration Solutions

Paul Hoffman, Stephen Hancock and Sanjay Chandru

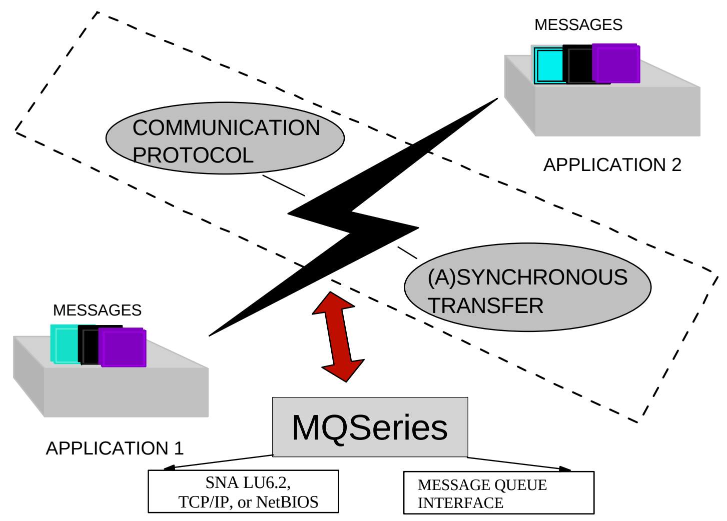
Introduction

VisualAge Generator® 4.5 simplifies the implementation of programs that communicate via messages by allowing the developer to access MQSeries® message queues using a high level file interface for putting messages on queues and getting messages from queues.

What is MQSeries Messaging?

MQSeries products enable programs to communicate with each other over a network of unlike components - processors, operating systems, subsystems, and communication protocols - using a simple and consistent application program interface (API) for putting messages on queues and getting messages off queues. Programs communicate via message queues so that the programs can run independently of each other at different speeds and times, in different locations, and without having a physical or logical connection between them.

FIGURE 1



MQSeries products ensure reliable message delivery and communication in a distributed computing environment. It simplifies application to application communication by by providing the MQSeries API which, without any additional programmer code, ensures the integrity of the transmitted data. Time independent processing is another major feature facilitated by the MQSeries product. This makes it an ideal tool for message processing in an asynchronous environment (Figure 1).

The MQSeries API is called the Message Queue Interface (MQI). An application programmer uses the MQI to request services from the queue manager. Typically calls would include putting a message onto a queue, getting a message from a queue, or triggering an application whenever a message arrives on a queue. Queues are managed by queue managers. These queue managers can manage several queues. If a message needs to be sent to a queue on a network, a Message Channel Agent (MCA) is used to send or receive messages between the two queue managers. The queue managers can communicate with each other using a standard communication protocol such as SNA or TCP/IP.

Since MQSeries manages the queues and transfering messages between the queues, the developer need not worry about the platform of the queue object to which this message needs to be sent on the network. MQSeries takes care of putting the message on the destination queue as long as a few standard message parameters are set to valid values. There are thirteen important MQI calls. Each of these calls consist of a set of parameters that need to be initialized. These parameters can be both input and output parameters. They contain instruction or data required by the queue manager to successfully process the call. Output parameters are set to indicate either success or failure with a reason code and a completion code.

An example of an MQSeries API call would be the MQOPEN call that is used to gain access to a queue:

MQOPEN(HConn, ObjDesc, Options, Hobj, CompCode, Reason)

where the individual parameters are defined as follows:

Hconn (Data Type:MQHConn): Input (Connection Handle)
ObjDesc (Data Type:MQOD): Input/Output (Object Descriptor)
Options (Data Type: MQLong): Output (Options)
Hobj (Data Type:MQHObj): Output (Object Handle)
CompCode (Data Type: MQLong): Output (Completion Code)
Reason (Data Type: MQLong): Output (Reason Code)

In the above example the MQOPEN call establishes an application's access to a queue. On a successful call, the object handle (Hobj) is returned, which is used in all subsequent calls to that queue object. The completion code and reason code are also set to indicate the success of the call.

MQSeries and VisualAge Generator Integration

So where does MQSeries fit in with the VisualAge Generator product? VisualAge Generator and MQSeries are complementary to each other. Just as MQSeries enables developers to communicate between applications in an easy and consistent way that is independent of the runtime platform, so VisualAge Generator enables developers to create applications - easily, consistently and independent of the runtime. The two products fit together to give a total solution for transaction processing utilizing synchronous or asynchronous message transfer.

VisualAge Generator:

- Is relatively simple to use.
- Is an excellent tool for rapid application deployment in an enterprise.
- Supports the distributed computing paradigm completely.
- Facilitates easy development and integration of applications on vastly dissimilar platforms.
- Cuts down application development time, thereby increasing productivity.

These features can be leveraged by MQSeries program developers to improve the ease of deployment, and save on resources. VisualAge Generator is an effective tool for hiding the complexities of implementing MQSeries functionality on a multi-platform environment. Furthermore, VisualAge Generator supports many of the platforms on which MQSeries can be currently deployed. As a result, there is a reduction in the complexity of design.

Enterprise client-server development for various platforms is the basis for the VisualAge Generator product. Thus MQSeries applications written on VisualAge Generator can result in an optimized development and deployment cycle. This gives an edge to any MQSeries developer, in terms of rapid and effective enterprise application deployment. With Version 4.5 MQI calls can be made from VisualAge Generator generated programs using a file level interface. These calls can be written in the VisualAge Generator 4GL without worrying about the complexities of the underlying platform and generated language. The use of VisualAge Generator to develop MQSeries applications would result in easier integration across multiple platforms. Thus the need to learn the intricacies of each platform on which MQSeries is deployed is greatly reduced.

An additional aspect of VisualAge Generator that makes it a very attractive tool for deploying MQSeries applications is its Interactive Test Facility (ITF). The applications developed for the above example can be tested separately before actually deploying them on a larger scale. This ensures bug free application development. MQSeries also fits in perfectly with the component based approach to application development in VisualAge Generator.

File Level Support for messaging in VisualAge Generator 4.5

VisualAge Generator 4.5 introduces support for MQ calls using VisualAge Generator file I/O options. The main benefit of this implementation is to abstract MQ API level calls so that the user need have no knowledge of API level MQ implementation. The implementation takes advantage of existing I/O options and EZE words. Basic MQSeries queue access requires little or no knowledge of MQ APIs to implement the calls (except a general understanding of message queueing). MQ functionality can be accessed using the ADD, SCAN and CLOSE VisualAge Generator I/O options. No new skills are required to use the new functions if you are already familiar with VisualAge Generator. All that is needed with this implementation is the name of the queue manager and queue where the message is to be sent. Access to advanced MQSeries functions is provided by allowing modifications to reusable parts shipped with VisualAge Generator 4.5 that contain options for accessing MQSeries APIs directly.

File level support includes:

- Automatic connection to queue manager
- Automatic opening of queues
- Automatic closing and disconnection
- Automatic return code checking
- Automatic data conversion
- Optional termination on hard errors
- Optional access to MQ control blocks
- Transaction control using EZECOMIT and EZEROLLB

To implement message queue access, the developer defines records with type MQMESSAGE and uses the ADD, SCAN, and CLOSE verbs to perform I/O operations with the messages.

The ADD I/O option connects to a queue manager, opens the queue and puts the message on the queue. The queue manager name or queue name are specified in the resource association file or set directly by the program in the EZEDEST special function word. Redundancy is eliminated by making a connection call only if there is no existing connection handle to the appropriate queue manager. A queue is opened for input or output only if it is not already open for the appropriate function. This greatly streamlines MQ access, making it more efficient and productive.

The SCAN I/O option gets a message from a queue based on the specified options. Like the ADD option, it automatially connects to the queue manager and opens the queue, if required.

The CLOSE I/O option closes any existing connections to specified queues after queue access. Resources are cleaned up when the program ends or where it is appropriate.

The developer can check the completion status of message I/O operations by testing the file states (no record found, soft error, hard error), or by checking the MQ completion and status codes in the EZERT2 and EZERT8 special function words.

Messages can be variable length. The developer specifies a record length item or a number of occurrences item within the record. The generated program specifies the correct length when calling the MQ APIs.

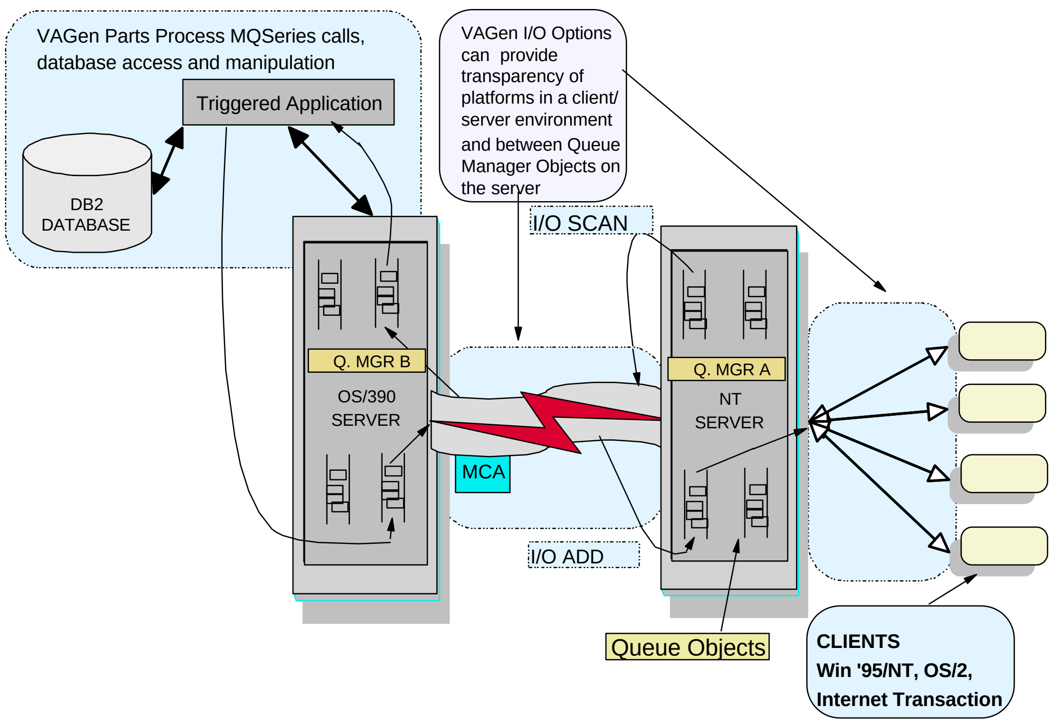
The developer can request data format conversion for the character and numeric items in the message by specifying a conversion table name in the resource association file or the EZECONVT special function word.

Messages can participate in units of work which are committed or rolled back by calling existing EZE words (EZECOMIT and EZEROLLB). The generated program uses the commit protocol appropriate for the target environment. The developer does not need to understand how transaction control is done in different platforms if the application is needed both in OS/390™ as well as AS/400™ or Windows NT. All the user needs to do is code an EZECOMIT or EZEROLLB call. VisualAge Generator takes care of implementing transaction control across all the different supported platforms.

MQSeries provides more than 150 different options that can be set for accessing queue, and queue manager functionality. These options are controlled through the setting of options parameters and control blocks passed on the MQOPEN, MQGET, and MQPUT APIs. Generated MQ programs build these parameters and control blocks based on the properties specified in the message record. If the developer wants the program to set values in these control blocks to control functions beyond those described above, the developer can specify the names of the MQ options records he wants to override in the advanced options record list in the message record properties.

Sample MQSeries Application Developed with VisualAge Generator 4.5 .

FIGURE 2


VAGen Parts Process MQSeries calls, database access and manipulation
Triggered Application
DB2 DATABASE
VAGen I/O Options can provide transparency of platforms in a client/ server environment and between Queue Manager Objects on the server
I/O SCAN
Q. MGR B
OS/390 SERVER
MCA
Q. MGR A
NT SERVER
I/O ADD
Queue Objects
CLIENTS
Win '95/NT, OS/2,
Internet Transaction

A sample MQSeries application demonstrates one type of system which developers can construct using VisualAge Generator 4.5 and the MQSeries products for the platforms involved. The following illustration contains client programs on various platforms, two server programs on two platforms, and a database program on one platform. Additionally, two queues are set up, one on each server platform

From three to five operating systems are involved running on workstation and mainframe hardware. The complexities of developing client and server programs on the Windows 95®, Windows NT®, and OS/390 platforms shown are handled by VisualAge Generator and MQSeries. A developer writes the client and server programs in the VisualAge Generator 4GL and generates client and server programs.

The two queues described are designed to allow a server to receive messages on one queue and put messages on the other. The two server programs, one on Windows NT and the other on OS/390, can pass messages in this manner while each monitoring just one queue for incoming messages.

Once the system is in production, the generated server program on Windows NT receives information from the client programs and puts messages onto a queue. Code generated by the developer's use of the ADD I/O option puts the message on the queue. The queue is actually running on the OS/390 operating system (a remote queue) although it is known by the Windows NT server program (through a local definition). The generated OS/390 server program gets the messages from the queue. Code generated by the developer's use of the SCAN I/O option gets the messages from the queue. The generated OS/390 server program passes the data using SQL commands to a DB2 database also running on the OS/390 operating system. VisualAge Generator is well suited to handle such transactions and SQL communications. Once operations involving the DB2 database are complete, any returning information is written into messages and put onto a queue running on the Windows NT operating system (a remote queue) known by the OS/390 server program (through a local definition). The Windows NT server gets the messages from the queue and sends information to the appropriate client programs, completing the transaction cycle.

The generated server programs can perform error handling and commit or rollback messages as needed in the course of system operations.

MQSeries triggering capabilities allow programs to be started when a message is put on a particular queue or shutdown when no new messages are placed on the queue during a specified length of time. If a server, such as the OS/390 in the example above, uses triggering, system resources can be saved by starting the server program only when the Windows NT server program puts a message on the OS/390 queue.

The sample application illustrates the diversity of programs and operating systems supported by VisualAge Generator. Other system configurations are possible, including different designs for the number and location of queues, queue managers, generated server programs, etc.

Prerequisites for Running Programs

- Install MQSeries server and client on the platform on which you wish to test the sample applications (MQSeries Version 5.x for AIX, AS/400, HP-UX, OS/2, Sun Solaris, Windows NT, Version 1.x for OS/390, Version 2.x for VSE).
- Verify queue manager using MQSeries sample programs provided with MQSeries software
- VisualAge Generator 4.5 needs to be installed and configured.
- Load the sample program .dat file and define queues to the Resource Association File
- Start the queue manager. The target queue must have its Get and Put message options enabled.
- Run the sample programs in test facility or generate and run in any supported runtime environment.

Information and Reference

For general information on MQSeries and VisualAge Generator:

Http://www-4.ibm.com/software/ts/mqseries/

Http://www-4.ibm.com/software/ad/visgen/

Http://www.redbooks.ibm.com/; MQSeries Primer, REDP0021

For specific information on implementing MQSeries programs in VisualAge Generator, refer to Chapter 4, Developing MQSeries Application Systems pp.77-pp.107 in the VisualAge Generator Version 4.5 User's Guide.

Trademarks