

IBM MQ V9 Application Development (Windows Labs)

WM513 (Classroom)

ZM513 (Self-paced)

Course description

This course helps you develop the skills that are necessary to implement various application requirements on IBM MQ versions up to and including IBM MQ V9.0.2. It focuses on procedural application development for IBM MQ.

The course begins by describing IBM MQ and the effect of design and development choices in the IBM MQ environment. It then covers IBM MQ application programming topics such as methods of putting and getting messages, identifying code that creates queue manager affinities, working with transactions, and uses of the publish/subscribe messaging style.

Finally, the course describes the IBM MQ Light interface, introduces Advanced Message Queuing Protocol (AMQP), and explains how to set up an AMQP channel and how to interface with IBM MQ Light.

Hands-on exercises throughout the course reinforce the lecture material and give you experience with IBM MQ clients.

For information about other related courses, see the IBM Training website:

http://www.ibm.com/training

General information

Delivery method

Classroom or self-paced virtual classroom (SPVC)

Course level

ERC 1.0

Product and version

IBM MQ V9

Audience

This course is designed for application developers and architects who are responsible for the development and design of IBM MQ applications.

Learning objectives

After completing this course, you should be able to:

* Describe key IBM MQ components and processes
* Explain the effect of design and development choices in the IBM MQ environment
* Describe common queue attributes and how to control these attributes in an application
* Differentiate between point-to-point and publish/subscribe messaging styles
* Describe the calls, structures, and elementary data types that compose the message queue interface
* Describe how IBM MQ determines the queue where messages are placed
* Explain how to code a program to get messages by either browsing or removing the message from the queue
* Describe how to handle data conversion across different platforms
* Explain how to put messages that have sequencing or queue manager affinities
* Explain how to commit or back out messages in a unit of work
* Describe how to code programs that run in an IBM MQ Client
* Explain the use of asynchronous messaging calls
* Describe the basics of writing publish/subscribe applications
* Describe the Advanced Message Queuing Protocol (AMQP)
* Differentiate among the various IBM MQ Light AMQP implementations
* Explain how to use IBM MQ applications to interface with IBM MQ Light

Prerequisites

* Successful completion of *Technical Introduction to IBM MQ* (WM103G), or comparable experience with IBM MQ
* Experience in business application design
* Experience in C language development

Duration

3 days

Skill level

Intermediate

Notes

The following unit and exercise durations are estimates, and might not reflect every class experience. If the course is customized or abbreviated, the duration of unchanged units will probably increase.

This course is an update of the following previous course:

* *IBM WebSphere MQ V8 Application Development (Windows Labs)* (WM507G)

Course agenda

|  |
| --- |
| Course introduction  Duration: 15 minutes |

|  |  |
| --- | --- |
| Unit 1. IBM MQ overview  Duration: 1 hour | |
| Overview | This unit provides an understanding of IBM MQ as a base to the development lectures, with an emphasis on writing well behaved, scalable applications. An IBM MQ developer can code programs that might result in performance problems, or can introduce affinities that impose limits on the ability to scale the infrastructure. This unit lays the foundation for the topics in subsequent units, and introduces potential issues to avoid. |
| Learning objectives | After completing this unit, you should be able to:   * Explain the advantages of message-oriented middleware * List the basic IBM MQ components * Describe the correct terminology to use when working with IBM MQ resources * Distinguish the various types of queues and how they are used * List basic IBM MQ application programming interface functions * Explain queue name resolution * Explain IBM MQ channels * Describe how application design affects IBM MQ clusters * Describe queue sharing groups * Describe the use of triggering in IBM MQ * Explain the differences between IBM MQ clients and IBM MQ servers * Distinguish between point-to-point and publish/subscribe messaging styles * Describe attributes that are present in a queue definition * Explain the message descriptor fields, how they relate to queue attributes, and how they influence application behavior * Distinguish between local and global units of work * Describe how design and development decisions affect various IBM MQ resources * Describe IBM MQ security and how it might affect application development * Explain where to look for information on IBM MQ errors |

|  |  |
| --- | --- |
| Exercise 1. Working with IBM MQ to find your message  Duration: 1 hour | |
| Overview | This exercise explains where and how to look for messages that you need to put or get with your code. It also describes the technology for which you are developing code. In addition, the unit shows you how to define various types of queues that can reduce your dependency on infrastructure services. |
| Learning objectives | After completing this exercise, you should be able to:   * Determine the status of queue managers in a server * Start a queue manager * Use the runmqsc utility and command scripts to create IBM MQ objects and check results * Put messages to local and alias queues and determine whether the messages arrived at the intended destination * Determine the trajectory and possible stops of a message put to a remote queue * Start a sender channel and check the channel status * Check the queue manager error logs * Determine where your message is * Examine the dead letter queue and identify the reason that a message was placed in the queue |

|  |  |
| --- | --- |
| Unit 2. Basic design and development concepts  Duration: 1 hour | |
| Overview | This unit introduces the components of the message queue interface, or MQI. You learn about header files, structures, and other items needed for your code. You alter a program to add processing of a second queue. You learn how the attributes you use in your code supersede object definition attributes. You learn about the MQCONNX, MQOPEN, MQPUT, MQCLOSE, and MQDISC calls. Finally, you learn how to determine the connection authentication settings of a queue manager, and how to incorporate connection authentication code in your program. |
| Learning objectives | After completing this unit, you should be able to:   * Describe common messaging patterns * Explain key architecture and performance considerations for message and application design * List the available programming options * Describe the calls, structures, and elementary data types that compose the message queue interface * Describe message types and message formats * Explain how to use the MQCONN or MQCONNX calls, and the various options of the MQCONNX call, to connect to a queue manager * Describe how the MQOPEN, MQPUT, and MQGET calls use the output of the MQCONN or MQCONNX calls * Explain how to use the MQCNO and MQCSP structures with the MQCONNX function call to implement connection authentication * Describe the use of the MQINQ and MQSET calls and the differences between them * Distinguish the superseding characteristics between object and MQI attributes * Describe how to compile a C program in the Linux and Windows environments |

|  |  |
| --- | --- |
| Exercise 2. Getting started with IBM MQ development  Duration: 1 hour | |
| Overview | In this exercise, you learn how to make basic changes and compile an MQI program. You use named constants to determine the superseding IBM MQ behavior when the same object attribute and MQI attribute use different values. You then experience the outcome of an incorrect version number in a structure, which you discover by learning how to code connection authentication in the MQCONNX call. You also learn how to check the queue manager connection authentication settings. |
| Learning objectives | After completing this exercise, you should be able to:   * Compile and test a copy of the put message sample program * Review the cmqc.h structure and the initialization values for the message descriptor structure * Review selected default values of a local queue definition * Add MQOPEN, MQPUT, and MQCLOSE calls to an application * Change persistence attributes in a program by using named constants * Determine the outcome of persistence behavior when the queue definition attributes and the MQI attributes use different values * Check the queue manager environment to determine the connection authentication settings * Set a variable to test connection authentication with a program * Determine the results of not setting the correct version number in a structure |

|  |  |
| --- | --- |
| Unit 3. MQOPEN, queue name resolution, and MQPUT  Duration: 1 hour | |
| Overview | This unit provides a detailed look at the MQOPEN and MQPUT calls. You learn how MQOPEN facilitates queue name resolution and the creation of dynamic queues. You also learn about the fields in the message descriptor structure, and how to use these fields in your application. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the details that the MQOPEN call handles * Identify the information in the object descriptor (MQOD) structure * Describe the options that can be specified in the MQOPEN call * Describe how the MQOPEN call processes queue name resolution * Explain the use of fields in the message descriptor (MQMD) structure * Describe how the IBM MQ V8.0.0.4 expiry cap overrides higher expiry specifications in the application * Describe various uses of Report messages * Describe the two types of context information and how context can be used to identify the user of an application * Examine use of the MQPUT1 call and identify optimal scenarios for its use * Explain how to create and remove temporary or permanent dynamic queues |

|  |  |
| --- | --- |
| Exercise 3. Working with MQOPEN and queue name resolution, MQPUT, and MQMD fields  Duration: 1 hour | |
| Overview | This exercise reinforces your knowledge of the MQOPEN and MQPUT calls. You learn about queue name resolution by coding a program to put a message to a remote queue manager without using a remote queue. You learn how to create dynamic queues with various naming options. You then learn how to use the Report field to request confirmation on arrival and Expiry messages. You also learn how to work with the IBM MQ expiry cap object attribute. |
| Learning objectives | After completing this exercise, you should be able to:   * Code various combinations of queue manager and queue name in the object descriptor to test and confirm how queue name resolution takes place * Code report options and review the results in the reply-to queue * Create and display a dynamic queue where the queue manager determines the name * Create and display a dynamic queue by specifying a partial prefix of the queue name * Create and display a dynamic queue by specifying the exact queue name * Request a confirm-on-arrival with data report message * Set the expiry attribute in a message and request an expiry report * Set the expiry attribute in a message for a queue with a lower expiry cap value in the queue definition |

|  |  |
| --- | --- |
| Unit 4. Getting messages and retrieval considerations  Duration: 1 hour | |
| Overview | This unit describes the various ways to retrieve messages from a queue. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the parameters that are required for the MQGET call * Describe the MQGET call option groupings * Explain how to associate requests with responses by using the message and correlation IDs * Differentiate between the options that are used to browse messages * Explain how to use message tokens to browse a queue * Explain the use and need for message marks and cooperative browsing * Describe how to write code that waits for responses |

|  |  |
| --- | --- |
| Exercise 4. Correlating requests to replies  Duration: 1 hour | |
| Overview | In this exercise, you learn how to work with the common task of correlating a reply message with a request message. |
| Learning objectives | After completing this exercise, you should be able to:   * Code or modify an application to generate a confirm-on-arrival (COA) Report message that preserves the original message identifier (MsgId) * Code or modify an application to reply to a request message by setting the correlation identifier of the reply message to the message identifier of the request message * Use a formatted message descriptor display to check the correct setting of your message and correlation identifier fields * Alter or code an application to get a reply message from a queue with a correlation identifier that matches the message identifier of its corresponding request message |

|  |  |
| --- | --- |
| Unit 5. Data conversion  Duration: 30 minutes | |
| Overview | This unit describes considerations to observe when data needs to be converted due to its exchange across different platforms. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the need for data conversion * Identify key MQMD data conversion fields * Differentiate how IBM MQ and message data are converted * Describe the various cases for message data conversion * Explain how to create a data conversion exit * Describe considerations to observe in the original MQPUT * Explain default data conversion * Identify the case when the sender handles conversion |

|  |  |
| --- | --- |
| Unit 6. Bind and Message groups  Duration: 30 minutes | |
| Overview | IBM MQ architects strive to design applications that are conducive to a highly available infrastructure. A key consideration in application design is avoidance of queue manager affinities. However, when queue manager affinities cannot be avoided, applications might need to use bind options and message groupings to accomplish the task. This unit shows you how to use bind options in clustered environments, and how to develop applications that need to produce or consume a group of messages in a specific order. |
| Learning objectives | After completing this unit, you should be able to:   * Explain the importance of limiting applications that introduce queue manager affinities in the IBM MQ architecture * Describe the use of bind-related attributes in queue definitions and MQOPEN options * Describe how to write IBM MQ applications that require a distinct sequence of messages to complete processing |

|  |  |
| --- | --- |
| Unit 7. Committing and backing out units of work  Duration: 30 minutes | |
| Overview | In this unit, you learn how to coordinate actions that must occur together to complete a valid process. The unit describes some of the terminology that is used, local and global units of work, and the IBM MQ function calls that are used for completion or back out of work. Finally, the unit describes details and considerations for using triggering and syncpoint. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the terminology that is associated with committing and backing out units of work * Differentiate between local and global units of work * Describe how syncpoint control is implemented in IBM MQ * Describe the syntax and use of the MQBEGIN, MQCMIT, and MQBACK function calls * Explain how to use triggering to start an application * Describe considerations to observe when using triggering and syncpoint in the same application |

|  |  |
| --- | --- |
| Exercise 5. Commit and back out review  Duration: 15 minutes | |
| Overview | This exercise reinforces the topics in the commit and back out unit by reviewing the sample program amqsxag0.c. |
| Learning objectives | After completing this exercise, you should be able to:   * Locate an example of a commit and back out application in the IBM MQ installation * Include the back out structure declaration and initialization in your application * Code the MQBEGIN function call * Design and code the flow of tasks that are required for a commit or back out application * Code the MQCMIT and MQBACK function calls |

|  |  |
| --- | --- |
| Unit 8. Asynchronous messaging  Duration: 30 minutes | |
| Overview | In this unit, you learn about IBM MQ callback and control functions that enable delivery of messages to a “unit of code” or module for consumption. This unit uses asynchronous messaging, which is accomplished with the IBM MQ MQCB and MQCTL function calls. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the concept and use cases for asynchronous messaging * Explain the parameters and operation options of the IBM MQ callback (MQCB) function call * Identify the fields in the callback data descriptor (MQCBD) structure * Describe the parameters and use of the IBM MQ control (MQCTL) function call * Identify the fields in the callback context (MQCBC) structure * Differentiate between the context of the MQCB\_FUNCTION and the MQCB function call |

|  |  |
| --- | --- |
| Exercise 6. Asynchronous messaging review  Duration: 30 minutes | |
| Overview | This exercise reinforces and extends the asynchronous messaging concepts and mechanics that are covered in the lecture. Rather than writing code, you follow the actions that are taken in sample program amqcbf0.c, and identify key portions of the process. You analyze the code and run the pre-compiled binary to see asynchronous messaging in action. |
| Learning objectives | After completing this exercise, you should be able to:   * Describe the mechanics and component exchanges of asynchronous messaging applications * Explain the role of the MQCB function call and its required parameters * Differentiate between the IBM MQ MQCB callback function and the application callback module that the MQCB\_FUNCTION parameter definition represents * Describe how to initiate and end consumption of messages with the MQCTL function call * Explain how to code and exchange information in an application callback function |

|  |  |
| --- | --- |
| Unit 9. IBM MQ clients  Duration: 30 minutes | |
| Overview | In your development work, you might need to code a program for an IBM MQ client environment. This unit explains the differences between IBM MQ servers and IBM MQ clients, and various ways to connect an IBM MQ client to an IBM MQ server. You also learn the difference to observe when compiling client code. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the differences between an IBM MQ client and an IBM MQ server * List the supported languages per platform that can be used to code an IBM MQ client application * Explain how to compile an IBM MQ client application * Describe the considerations to observe when working with IBM MQ clients * Describe the use of environment variables or a client channel definition table (CCDT) to connect an IBM MQ client to a queue manager in an IBM MQ server * Explain how to use the MQCONNX call to connect an IBM MQ client application to a queue manager directly from your code * Differentiate the capabilities that are available with the various client connectivity options * Describe a redistributable client (V8.0.0.4 and later) * Summarize the use and precedence order of a CCDT URL |

|  |  |
| --- | --- |
| Exercise 7. Working with an IBM MQ client  Duration: 1 hour | |
| Overview | This exercise shows you how to compile a program with the IBM MQ client libraries. You also learn how to connect the IBM MQ client to the queue manager in three different ways, which types of connections supersede others, and how to connect with code without configuring the IBM MQ client. |
| Learning objectives | After completing this exercise, you should be able to:   * Compile an existing program as an IBM MQ client * Review a client connection channel definition and test connectivity to a queue manager by using the client channel definition table * Configure and test connectivity to the queue manager by using first the MQCCDTURL and then the MQSERVER environment variable * Alter the client program to use the MQCONNX call to establish connectivity to the queue manager |

|  |  |
| --- | --- |
| Unit 10. Introduction to publish/subscribe  Duration: 1 hour | |
| Overview | This unit teaches you how to work with the publish/subscribe messaging style. It explains the history of publish/subscribe, describes its components, and defines key terminology to use when referring to the co-existing publish/subscribe capabilities. The unit also teaches you how to code subscriber and publisher applications. |
| Learning objectives | After completing this unit, you should be able to:   * Differentiate between publish/subscribe and point-to-point messaging * Describe how the history of publish/subscribe influences its functions and terminology * Identify the basic components of publish/subscribe * Describe key properties of topics, subscriptions, and publications * Explain how to write an application that subscribes to a topic by using the MQSUB function call * Describe how to code the MQOPEN and MQPUT calls to write an application that publishes to a topic |

|  |  |
| --- | --- |
| Exercise 8. Working with publish/subscribe basics  Duration: 45 minutes | |
| Overview | In this exercise, you learn how to use the integrated publish/subscribe API to complete the code in a subscriber application. As part of the exercise, you convert the putmsg.c program to publish messages to a topic instead of putting messages to a queue. |
| Learning objectives | After completing this exercise, you should be able to:   * Add code to an application to use the subscription descriptor and the MQSUB function call to create a subscription * Explain how to pass the managed queue handle from the MQSUB function call to access the managed queue in a subsequent MQGET function call * Convert an application that puts messages to a queue to publish messages to a topic * Use IBM MQ Explorer to obtain more details about your subscription |

|  |  |
| --- | --- |
| Unit 11. Advanced Message Queuing Protocol (AMQP) and IBM MQ Light  Duration: 1 hour | |
| Overview | In this unit, you learn about the Advanced Message Queuing Protocol (AMQP), IBM MQ Light, and how to exchange messages with IBM MQ. You also learn about the node.js IBM MQ Light client implementation. Next, you use what you learned about IBM MQ Light and IBM MQ publish/subscribe to exchange messages between IBM MQ and IBM MQ Light. The unit includes mapping considerations between IBM MQ and IBM MQ Light, and how to enable an IBM MQ queue manager to use AMQP channels. You also learn how to use IBM MQ publish/subscribe sample programs to test the exchange of messages between a queue manager and IBM MQ Light. |
| Learning objectives | After completing this unit, you should be able to:   * Describe the Advanced Message Queuing Protocol * Describe typical IBM MQ application and IBM MQ Light application interface scenarios * Describe basic IBM MQ Light concepts and list the components * Describe the quality of service categories available with IBM MQ Light * Describe how to write code to send and receive messages between IBM MQ applications and IBM MQ Light node.js applications * Describe how to map headers and properties between IBM MQ applications and AMQP applications * Explain how to enable and configure IBM MQ – IBM MQ Light interface * Describe the commands that are used to check the IBM MQ AMQP channel connections * Explain where to locate the logs that hold IBM MQ Light-related information * Describe the security options for IBM MQ Light |

|  |  |
| --- | --- |
| Exercise 9. Connecting IBM MQ Light applications to IBM MQ applications  Duration: 1 hour | |
| Overview | This exercise gives you hands-on experience with connecting IBM MQ Light applications to IBM MQ applications. You review the pre-configured AMQP-IBM MQ Light environment, and start the AMQP service and channel. You then use the queue manager as an IBM MQ Light messaging provider, and exchange messages between IBM MQ and IBM MQ Light. |
| Learning objectives | After completing this exercise, you should be able to:   * Examine the IBM MQ Light components that are configured in the queue manager * Start and check the status of the AMQP service * Start and check the status of an AMQP channel * Use an IBM MQ Light application and the sample node.js IBM MQ Light client application to subscribe to a topic of interest * Use an IBM MQ Light application and the sample node.js IBM MQ Light client application to publish messages to interested subscribers * Use a sample IBM MQ publish/subscribe application to publish messages of interest to subscribed IBM MQ and IBM MQ Light node.js applications * Use a sample IBM MQ publish/subscribe application to subscribe to messages of interest that might proceed from IBM MQ applications or IBM MQ Light publish applications * Examine the IBM MQ objects that are necessary for a queue-based application to receive messages of interest from an IBM MQ Light message producer application * Use a sample IBM MQ application to get messages from a queue that an IBM MQ Light application publishes * Examine the IBM MQ object that is necessary for a queue-based application to put messages on a queue that can go to an interested IBM MQ Light client application * Use a sample IBM MQ application to put messages to a queue destined to interested IBM MQ Light subscriber applications |

|  |  |
| --- | --- |
| Unit 12. Course summary  Duration: 15 minutes | |
| Overview | This unit summarizes the course and provides information for future study. |
| Learning objectives | After completing this unit, you should be able to:   * Explain how the course met its learning objectives * Access the IBM Training website * Identify other IBM Training courses that are related to this topic * Locate appropriate resources for further study |

For more information

To learn more about this course and other related offerings, and to schedule training, see **ibm.com**/training

To learn more about validating your technical skills with IBM certification, see **ibm.com**/certify