IBM WebSphere MQ (MQ Series) in a Nutshell

(Prepared by Channu Kambalyal)

Introduction

- MQ Series is a Middleware for Commercial Messaging and Queuing
- MQ Series API – also called Message Queue Interface (MQI) used to communicate with a Message Queue Manager (MQM), the runtime program of MQSeries.
- MQI consists of 13 calls.

Messages

- Consists of 2 parts: Message Descriptor and Message Data
- Message descriptor identifies message id and contains control information such as message type, expiry time, correlation ID, priority, reply queue name.
- MQ Series Version 5 supports maximum message length of 100 MB
- Messages can be segmented or grouped. If permitted queue manager segments a large message when it does not fit in a queue. Application has the option to receive entire message in piece or each segment separately. Application can also control the logical boundaries or buffer size of the segment of a message. Queue Manager ensures that the order of the segment is maintained.
- Several small messages can be grouped to build one larger physical message.
- Using a Distribution List, you can send a message to more than one destination. Distribution List is a file contains a list of queue names and queue managers that own them. Receiving Queue Manager replicates the messages and puts them into destination queues (This function is called late fan-out).
- MQ Series knows 4 types of massages: Datagram (unsolicited message), Request (a message for which response is expected), Reply and Report (an event such as occurrence of an error or confirmation on arrival or delivery)
- MQ Series messages can be persistent and non-persistent.
- Message Descriptor contains: Version, Message ID/Correlation ID, Persistent/ Non-Persistent, Priority, Date &m Time, Lifetime of a message, Return Address, Format, Sender Application and Type, Report options/ Feedback (COA, COD), Back out counter, Segmenting/ Grouping Information.

Queue Manager (MQM)

- Manages queues and messages for applications
- Transfers messages to other Queue Managers via channels using existing network facilities.
- It refers to objects that are defined by the administrator.
- Coordinates updates to databases and queues using two-phase commit.
- Gets and puts from/to queues are committed together with SQL updates, or backed out if necessary.
- Segments messages, if necessary, and assembles them.
- Can group messages and send them as one physical message to their destination, where they are automatically disassembled.
- Can send one message to more than one destination using a user-defined dynamic destination list.
- Allow administrators to create and delete queues, alter properties of existing queues, control the operation of queue manager.
- MQ Series for Windows NT version 5.1 provides GUI to administer.
- MQ Series for Windows (different from NT) is a single-user queue manager and is not intended to function as a queue manager for other MQ Series clients.
Queue Manager Clusters

- MQSeries for MVS/ESA and Version 5.1 for distributed platforms, provides clustering features.
- Queue Managers that form a cluster can run in the same machine or in different machines on different platforms.
- Two of Queue Managers maintain a repository that contains information about all queue managers and queues in the cluster (full repository). Other Queue Managers maintain only a repository of the objects they are interested in (partial repository).
- Queue Managers use special cluster channels to exchange information.
- Client application may specify a queue manager and direct the message to a specific queue in a cluster or it may let a queue manager to determine where the queue is and to which one to send the message.
- Client uses a Transmission Queue on its machine and destination queue is called “Target Queue”.
- Administrator must define the name of the cluster, when a queue is defined.
- MQSeries distributes the messages round robin.

Queue Manager Objects

- A Queue Manager uses 3 types of objects, namely, Queues, Process Definitions and Channels.
- Queues are used to store messages.
- Process Definition object defines an application to a queue manager. It contains a name of a program (and its path) to be triggered when a message arrives for it.
- Channel is a communication link. There are 2 kinds of channels, namely, Message Channels and MQI channels.
- **Message Channel** connects 2 queue managers via Message Channel Agents (MCA). Message Channel is unidirectional.
- MCA is a program (also called mover) that transfers messages from a transmission queue to a communication link and from communication link to a target queue.
- **MQI channel** connects MQSeries client to a queue manager and is bi-directional.
- Message Channel can run at 2 speeds: fast and normal. Fast Channels improve performance but messages can be lost in case of channel failure.

Message Queues

- Message Queues belong to Queue Manager.
- Types of Messages Queues are:
  - Local Queue – is a real queue!
  - Cluster Queue – is a local queue that is known throughout a cluster of queue managers.
  - Remote Queue – structure describing a queue hosted by a different queue manager.
  - Transmission Queue – a local queue used for messages to be sent to a remote queue.
  - Initiation Queue – local queue with a special purpose
  - Dynamic Queue – local queue created on the fly
  - Alias Queue -- if you do not like the queue name
  - Dead-Letter Queue – one for each queue manager
  - Reply-to Queue – specified in request message
  - Model Queue – model for local queues
  - Repository Queue – hold cluster information
- Create a queue manager using the command: `crtmqm`. Example:
  - `Crtmqm /q /u system.dead.letter.queue MYQMGR`
- To start a queue manager issue command: `strmqm`
Manipulating Queue Manager Objects

- Use the utility RUNMQSC to manipulate queue manager objects. Ensure queue manager is started prior using the runmqsc utility. Example:
  - C:\strmqm
  - runmqsc
define qlocal('QUEUE1') replace descry ('test queue')
  - alter qmgr deadq(system.dead.letter.queue)
  - end

How MQSeries Works

Following diagram depicts how MQSeries works:

![Diagram of MQSeries](image)

**Figure: How MQSeries Works**

Communication between Queue managers

How to Trigger Applications

Message Queuing Interface

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCONN</td>
<td>Connect to a Queue Manager</td>
</tr>
<tr>
<td>MQDISC</td>
<td>Disconnect from a Queue Manager</td>
</tr>
<tr>
<td>MQOPEN</td>
<td>Open a specific queue</td>
</tr>
<tr>
<td>MQCLOSE</td>
<td>Close a queue</td>
</tr>
<tr>
<td>MQPUT</td>
<td>Put message on a queue</td>
</tr>
<tr>
<td>MQPUT1</td>
<td>Get message from a queue</td>
</tr>
<tr>
<td>MQGET</td>
<td>MQOPEN + MQPUT + MQCLOSE</td>
</tr>
<tr>
<td>MQINQ</td>
<td>Inquire properties of an object</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>MQSET</td>
<td>Set properties of an object</td>
</tr>
<tr>
<td>MQCONNX</td>
<td>Standard or fast path bindings</td>
</tr>
<tr>
<td>MQBEGIN</td>
<td>Begin a unit of work (database coordination)</td>
</tr>
<tr>
<td>MQCMIT</td>
<td>Commit a unit of work</td>
</tr>
<tr>
<td>MQBACK</td>
<td>Back out</td>
</tr>
</tbody>
</table>