Introduction to the MQI

Morag Hughson – morag@mqgem.com
MQGem Software

Agenda

- MQI Concepts
- MQI Structures & Datatypes
- Basic MQI walkthrough
  - With Demonstrations
  - A number of verbs we do not cover
    - MQCMIT, MQBACK, MQINQ, MQSET etc

MQI Calls

QMGR
Languages

- **Procedural (MQI)**
  - C
  - COBOL
  - Visual Basic
  - RPG
  - PL/1
  - Assembler
  - TAL

- **Object-Oriented (Classes)**
  - Java
  - JMS/XMS
  - C++
  - .NET languages
  - ActiveX (MQAX)
  - Perl

Interface

- **Simple ‘handle’ based interface**
  - Returned handle passed to subsequent call

- **Each verb returns**
  - Completion Code
    - MQCC-OK
    - MQCC-WARNING
    - MQCC-FAILED
  - Reason Code
    - MQRC-xxxxxxxx
    - MQRC-NONE

- Make sure you check the reason codes!
Data Structures

Programmers should be familiar with:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQMD</td>
<td>Message Descriptor</td>
<td>Attributes associated with a message</td>
</tr>
<tr>
<td>MQOD</td>
<td>Object Descriptor</td>
<td>Describes what object to open</td>
</tr>
<tr>
<td>MQSD</td>
<td>Subscription Descriptor</td>
<td>Describes what to subscribe to</td>
</tr>
<tr>
<td>MQPMO</td>
<td>Put Message Options</td>
<td>Describes how a message should be put</td>
</tr>
<tr>
<td>MQGMO</td>
<td>Get Message Options</td>
<td>Describes how a message should be got</td>
</tr>
</tbody>
</table>

Data Structure Tips

- Use copybooks with value initialisers
  ▶ 01 MESSAGE-DESCRIPTOR.
  ▶ COPY CMQMDV.
  ▶ Initialise to version 1

- Structures are versioned
  ▶ Set the minimum version you need
    - MOVE MQMD-VERSION-2 to MQMD-VERSION.
  ▶ Don't use current version
    - MOVE MQMD-CURRENT-VERSION to MQMD-VERSION.

- Bear in mind that some structures are input/output
  ▶ May need to reset values for subsequent call
    - Eg. MsgId & CorrelId fields of MQMD on MQGET call
MQ Elementary Data Types

- The main MQI data types

<table>
<thead>
<tr>
<th>DataType</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQHCONN</td>
<td>4-byte Connection Handle</td>
</tr>
<tr>
<td>MQHOBJ</td>
<td>4-byte Object Handle</td>
</tr>
<tr>
<td>MQLONG</td>
<td>4-byte binary integer</td>
</tr>
<tr>
<td>MQPTR</td>
<td>Pointer</td>
</tr>
<tr>
<td>MQCHARn</td>
<td>A series of “n” bytes containing character data</td>
</tr>
<tr>
<td>MQBYTEEn</td>
<td>A series of “n” bytes containing binary data</td>
</tr>
<tr>
<td>MQCHARV</td>
<td>Variable length string</td>
</tr>
</tbody>
</table>

Connect

MQCONN

Queue Manager Name

Connection Handle

Completion Code

Reason Code

Basic connect

QMGR
Connect with extended options

 MQCONNX

- Queue Manager Name
- Connection Options
- Connection Handle
- Completion Code
- Reason Code

- Handle sharing options
- Client channel specification
- FASTPATH connection
- Additional security settings
- Reconnect option

QMGR

Connecting

- MQCONNX
  - Don't hardcode QM name
  - Always check reason codes

- Connections options
  - Connection not thread specific
  - Client reconnect

CALL 'MQCONNX'
USING QM-NAME, CONNECTION-OPTIONS, HCONN, MQCC, REASON.

IF MQCC IS NOT EQUAL TO MQCC-OK
* report reason and
* stop if it failed
END-IF.

01 HCONN PIC S9(9) BINARY.
01 CONNECTION-OPTIONS.
COPY CMQCN0V.

ADD MQCNO-HANDLE-SHARE-BLOCK MQCNO-RECONNECT
GIVING MQCNO-OPTIONS.
MQCONN(X) Tips

- Don’t hardcode Queue Manager names
  - Pass as parameter or configure in INI file

- Best to use MQCONNX
  - Has options structure should it be needed

- Most expensive verb
  - Don’t issue it repeatedly for each request
    - Often problem for OO languages

- If MQI handle need to be used on different threads
  - Use connection options to indicate the MQI handle can be shared
  - Choose to block or reject any calls from another thread when handle is in use

- If reconnecting use exponential back-off with random wait
  - Try to avoid client storms

- Can dynamically load MQ libraries if client or local binding
  - Preferable to shipping two versions of the program

Open a Queue

- MQCONNX
- MQOPEN

Connection Handle
Object Options
Object Descriptor

Object Handle
Completion Code
Reason Code

- Indicate type of open required
  - input, output, inquire etc

- Indicate object name to open
  - Queue name
  - Topic
Open a queue

- **MQOPEN** a queue
- **OpenOptions**
  - MQOO- flags which are required
- **MQOD describes a object to open**
  - MQOD-OBJECTTYPE
    - MQOT-Q for point-to-point
    - MQOT-TOPIC for publish
  - MQOD-OBJECTSTRING
  - MQOD-OBJECTNAME

```assembly
01 Q-HANDLE PIC S9(9) BINARY.
01 OBJECT-DESCRIPTOR.
COPY CMQODV.

MOVE MQOT-Q TO MQOD-OBJECTTYPE.
MOVE QNAME TO MQOD-OBJECTNAME.
```

**Object Descriptor (MQOD)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
<td>1</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
<td></td>
</tr>
<tr>
<td>ObjectType</td>
<td>Object type</td>
<td></td>
</tr>
<tr>
<td>ObjectName</td>
<td>Object name</td>
<td></td>
</tr>
<tr>
<td>ObjectQMgrName</td>
<td>Object queue manager name</td>
<td></td>
</tr>
<tr>
<td>DynamicQName</td>
<td>Dynamic queue name</td>
<td></td>
</tr>
<tr>
<td>AlternateUserId</td>
<td>Alternate user identifier</td>
<td></td>
</tr>
<tr>
<td>RecsPresent</td>
<td>Number of object records present</td>
<td></td>
</tr>
<tr>
<td>KnownDestCount</td>
<td>Number of local queues opened successfully</td>
<td>2</td>
</tr>
<tr>
<td>UnknownDestCount</td>
<td>Number of remote queues opened successfully</td>
<td></td>
</tr>
<tr>
<td>InvalidDestCount</td>
<td>Number of queues that failed to open</td>
<td></td>
</tr>
<tr>
<td>ObjectRecOffset</td>
<td>Offset of first object record from start of MQOD</td>
<td></td>
</tr>
<tr>
<td>ResponseRecOffset</td>
<td>Offset of first response record from start of MQOD</td>
<td></td>
</tr>
<tr>
<td>ObjectRecPtr</td>
<td>Address of first object record</td>
<td></td>
</tr>
<tr>
<td>ResponseRecPtr</td>
<td>Address of first response record</td>
<td></td>
</tr>
<tr>
<td>AlternateSecurityId</td>
<td>Alternate security identifier</td>
<td></td>
</tr>
<tr>
<td>ResolvedQName</td>
<td>Resolved queue name</td>
<td>3</td>
</tr>
<tr>
<td>ResolvedQMgrName</td>
<td>Resolved queue manager name</td>
<td></td>
</tr>
<tr>
<td>ObjectString</td>
<td>Long object name</td>
<td></td>
</tr>
<tr>
<td>SelectionString</td>
<td>Selection string</td>
<td></td>
</tr>
<tr>
<td>ResolvedQName</td>
<td>Resolved long object name</td>
<td></td>
</tr>
<tr>
<td>ResolvedType</td>
<td>Resolved object type</td>
<td>4</td>
</tr>
</tbody>
</table>
Open Options

- Options can be added together as required

```
10 MQOO-BIND-AS-Q-DEF PIC S9(9) BINARY VALUE 0.
10 MQOO-READ-AHEAD-AS-Q-DEF PIC S9(9) BINARY VALUE 0.
10 MQOO-INPUT-AS-Q-DEF PIC S9(9) BINARY VALUE 1.
10 MQOO-INPUT-EXCLUSIVE PIC S9(9) BINARY VALUE 4.
10 MQOO-BROWSE PIC S9(9) BINARY VALUE 8.
10 MQOO-OUTPUT PIC S9(9) BINARY VALUE 16.
10 MQOO-INQUIRE PIC S9(9) BINARY VALUE 32.
10 MQOO-SET PIC S9(9) BINARY VALUE 64.
10 MQOO-SAVE-ALL-CONTEXT PIC S9(9) BINARY VALUE 128.
10 MQOO-PASS-IDENTITY-CONTEXT PIC S9(9) BINARY VALUE 256.
10 MQOO-SET-IDENTITY-CONTEXT PIC S9(9) BINARY VALUE 1024.
10 MQOO-ALTERNATE-USER-AUTHORITY PIC S9(9) BINARY VALUE 4096.
10 MQOO-FAIL-IF-QUIESCING PIC S9(9) BINARY VALUE 8192.
10 MQOO-BIND-ON-OPEN PIC S9(9) BINARY VALUE 16384.
10 MQOO-BIND-NOT-FIXED PIC S9(9) BINARY VALUE 32768.
10 MQOO-CO-OP PIC S9(9) BINARY VALUE 131072.
10 MQOO-NO-READ-AHEAD PIC S9(9) BINARY VALUE 524288.
```

MQOPEN Tips

- Try not to hardcode queue/topic names
- Try not to open queues exclusively
  - Will reduce options for workload balancing
- Use MQPUT1 if only opening queue to put one message
- Consider queue cache for common used queues
  - MQOPEN is relatively expensive – load and security check
- Use read ahead for performance gain
  - If client and non-persistent messaging
- If opening model reply queues
  - Be aware of how many instances of queues you may be creating
    - Particularly large numbers of clients.
  - May be better to share reply queue
Put a message

MQCONNX MQOPEN

MQPUT

QMGR

Connection Handle
Object Handle
Message Handle
Message Descriptor
Put Message Options
Message Data

Completion Code
Reason Code

- Updates structure
  - Message Descriptor
  - Put Message Options

Putting Application

- MQOPEN a queue
- MQPUT a message
  - Simple Hello World message
  - Set message format to string
  - Out of syncpoint

```
ADD MQOO-OUTPUT
MQOO-FAIL-IF-QUIESCING
GIVING OPTIONS.
CALL 'MQOPEN'
USING HCONN,
OBJECT-DESCRIPTOR,
OPTIONS,
Q-HANDLE,
MQCC,
REASON.
CALL 'MQPUT'
USING HCONN,
Q-HANDLE,
MESSAGE-DESCRIPTOR,
PMOPTIONS,
BUFFER-LENGTH,
BUFFER,
MQCC,
REASON.
```

```
01 MESSAGE-DESCRIPTOR.
  COPY CMQMDV.
01 PMOPTIONS.
  COPY CMQPMOV.
MOVE MQFMT-STRING TO MQMD-FORMAT.
ADD MQPMO-FAIL-IF-QUIESCING MQPMO-NO-SYNCPOINT GIVING MQPMO-OPTIONS.
MOVE 'Hello World!' TO BUFFER.
MOVE 12 TO BUFFER-LENGTH.
```
### Message Descriptor (MQMD)

<table>
<thead>
<tr>
<th>Field (V1)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
</tr>
<tr>
<td>Report</td>
<td>Options for report messages</td>
</tr>
<tr>
<td>MsgType</td>
<td>Message Type</td>
</tr>
<tr>
<td>Encypt</td>
<td>Message priority</td>
</tr>
<tr>
<td>Feedback</td>
<td>Message persistence</td>
</tr>
<tr>
<td>Encoding</td>
<td>Original length of message</td>
</tr>
<tr>
<td>CodedCharSet</td>
<td>Character set identifier</td>
</tr>
<tr>
<td>Format</td>
<td>Format name of message data</td>
</tr>
<tr>
<td>Priority</td>
<td>Message priority</td>
</tr>
<tr>
<td>Feedback</td>
<td>Feedback or reason code</td>
</tr>
<tr>
<td>Len</td>
<td>Message flag</td>
</tr>
<tr>
<td>CorrelId</td>
<td>Correlation identifier</td>
</tr>
<tr>
<td>ReplyToQ</td>
<td>Name of reply queue</td>
</tr>
<tr>
<td>ReplyToQMgr</td>
<td>Name of reply queue manager</td>
</tr>
<tr>
<td>UserIdentifier</td>
<td>User identifier</td>
</tr>
<tr>
<td>AccountingToken</td>
<td>Accounting token</td>
</tr>
<tr>
<td>ApplIdentityData</td>
<td>Application data relating to identity</td>
</tr>
<tr>
<td>PutApplType</td>
<td>Type of application that put the message</td>
</tr>
<tr>
<td>PutApplName</td>
<td>Name of application that put the message</td>
</tr>
<tr>
<td>PutDate</td>
<td>Date when message was put</td>
</tr>
<tr>
<td>PutTime</td>
<td>Time when message was put</td>
</tr>
<tr>
<td>ApplOriginData</td>
<td>Application data relating to origin</td>
</tr>
</tbody>
</table>

### Field (V2)

<table>
<thead>
<tr>
<th>Field (V2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupId</td>
<td>Group identifier</td>
</tr>
<tr>
<td>MsgSeqNumber</td>
<td>Sequence number of logical message within group</td>
</tr>
<tr>
<td>Offset</td>
<td>Offset of data in physical message from start of logical message</td>
</tr>
<tr>
<td>MsgFlag</td>
<td>Message flag</td>
</tr>
</tbody>
</table>

### Put Message Options (MQPMO)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
</tr>
<tr>
<td>Options</td>
<td>Options that control the actions of MQPUT and MQPUT1</td>
</tr>
<tr>
<td>Context</td>
<td>Object handle of input queue</td>
</tr>
<tr>
<td>KnownDestCount</td>
<td>Number of messages sent successfully to local queues</td>
</tr>
<tr>
<td>UnknownDestCount</td>
<td>Number of messages sent successfully to remote queues</td>
</tr>
<tr>
<td>InvalidDestCount</td>
<td>Number of messages that could not be sent</td>
</tr>
<tr>
<td>ResolvedQName</td>
<td>Resolved name of destination queue</td>
</tr>
<tr>
<td>ResolvedQMgrName</td>
<td>Resolved name of destination queue manager</td>
</tr>
<tr>
<td>RecsPresent</td>
<td>Number of put message records or response records present</td>
</tr>
<tr>
<td>PutMsgRecFields</td>
<td>Flags indicating which MQPMR fields are present</td>
</tr>
<tr>
<td>PutMsgRecOffset</td>
<td>Offset of first put message record from start of MQPMO</td>
</tr>
<tr>
<td>ResponseRecOffset</td>
<td>Offset of first response record from start of MQPMO</td>
</tr>
<tr>
<td>PutMsgRecPtr</td>
<td>Address of first put message record</td>
</tr>
<tr>
<td>ResponseRecPtr</td>
<td>Address of first response record</td>
</tr>
<tr>
<td>OriginalMsgHandle</td>
<td>Original message handle</td>
</tr>
<tr>
<td>NewMsgHandle</td>
<td>New message handle</td>
</tr>
<tr>
<td>Action</td>
<td>Type of put being performed and the relationship between the original message and the new message</td>
</tr>
<tr>
<td>PubLevel</td>
<td>Level of subscription targeted by the publication</td>
</tr>
</tbody>
</table>
Put Options

- Options can be added together as required

```
0  MQPMO-SYNCPOINT                  PIC S9(9) BINARY VALUE 2.
10 MQPMO-NO-SYNCPOINT               PIC S9(9) BINARY VALUE 4.
10 MQPMO-DEFAULT-CONTEXT            PIC S9(9) BINARY VALUE 32.
10 MQPMO-NEW-MSG-ID                 PIC S9(9) BINARY VALUE 64.
10 MQPMO-NEW-CORREL-ID              PIC S9(9) BINARY VALUE 128.
10 MQPMO-FAIL-IDENTITY-CONTEXT      PIC S9(9) BINARY VALUE 256.
10 MQPMO-FAIL-IF-QUIESCING          PIC S9(9) BINARY VALUE 8192.
10 MQPMO-NO-CONTEXT                 PIC S9(9) BINARY VALUE 16384.
10 MQPMO-ASYNC-RESPONSE             PIC S9(9) BINARY VALUE 65536.
10 MQPMO-SYNC-RESPONSE              PIC S9(9) BINARY VALUE 131072.
10 MQPMO-LOGICAL-ORDER              PIC S9(9) BINARY VALUE 4096.
10 MQPMO-ASYNC-RESPONSE             PIC S9(9) BINARY VALUE 65536.
10 MQPMO-SYNC-RESPONSE              PIC S9(9) BINARY VALUE 131072.
10 MQPMO-RESOLVE-LOCAL-Q            PIC S9(9) BINARY VALUE 262144.
10 MQPMO-FAIL-IF-QUIESCING          PIC S9(9) BINARY VALUE 16384.
10 MQPMO-LOGICAL-ORDER              PIC S9(9) BINARY VALUE 32768.
10 MQPMO-ASYNCH-RESPONSE            PIC S9(9) BINARY VALUE 65536.
10 MQPMO-ASYNC-RESPONSE             PIC S9(9) BINARY VALUE 131072.
10 MQPMO-RESOLVE-LOCAL-Q            PIC S9(9) BINARY VALUE 262144.
10 MQPMO-FAIL-IF-QUIESCING          PIC S9(9) BINARY VALUE 16384.
10 MQPMO-LOGICAL-ORDER              PIC S9(9) BINARY VALUE 32768.
10 MQPMO-ASYNCH-RESPONSE            PIC S9(9) BINARY VALUE 65536.
10 MQPMO-ASYNC-RESPONSE             PIC S9(9) BINARY VALUE 131072.
10 MQPMO-RESOLVE-LOCAL-Q            PIC S9(9) BINARY VALUE 262144.
```

MQPUT Tips

- Always use explicit syncpoint setting
  - Defaults are not the same on z/OS and Distributed
  - Generally
    - Syncpoint when persistent
    - No syncpoint when non-persistent
- Try not to use extreme message sizes
  - QM optimized for message 4K – 1MB
- Consider async put response for performance gain
  - If on client and sending many non-persistent messages
Get a message

MQCONNX MQOPEN MQPUT
MQOPEN MQGET

Connection Handle
Object Handle
Message Descriptor
Get Message Options
Buffer Size

Message Data
Message Length
Completion Code
Reason Code

Updates structure
- Message Descriptor
- Get Message Options

Getting Application

- MQOPEN a queue
- MQGET a message
  - Syncpoint if persistent
  - Always ask for convert
  - Wait for message
    - up to one minute

ADD MQOO-INPUT-SHARED
MQOO-FAIL-IF-QUIESCING
GIVING OPTIONS.

CALL 'MQOPEN'
USING HCONN,
  OBJECT-DESCRIPTOR,
  OPTIONS,
  Q-HANDLE,
  MQCC,
  REASON.

CALL 'MQGET'
USING HCONN,
  Q-HANDLE,
  MESSAGE-DESCRIPTOR,
  GMOPTIONS,
  BUFFER-LENGTH,
  BUFFER,
  DATA-LENGTH,
  MQCC,
  REASON.

01 MESSAGE-DESCRIPTOR.
  COPY CMQMDV.
01 GMOPTIONS.
  COPY CMQGMOV.
MOVE 60000 TO MQGMO-WAITINTERVAL.
ADD MQGMO-SYNCPNT-IF-PERSISTENT MQGMO-CONVERT
  MQGMO-FAIL-IF-QUIESCING MQGMO-WAIT GIVING MQGMO-OPTIONS.
## Get Message Options (MQGMO)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td>Options that control the action of MQGET</td>
<td></td>
</tr>
<tr>
<td>WaitInterval</td>
<td>Wait interval</td>
<td></td>
</tr>
<tr>
<td>Signal</td>
<td>Signal</td>
<td></td>
</tr>
<tr>
<td>Signal2</td>
<td>Signal identifier</td>
<td></td>
</tr>
<tr>
<td>ReceivedQName</td>
<td>Received name of destination queue</td>
<td></td>
</tr>
<tr>
<td>WatchOptions</td>
<td>Options controlling selection criteria used for MQGET</td>
<td></td>
</tr>
<tr>
<td>GroupStatus</td>
<td>Flag indicating whether message retrieved is in a group</td>
<td></td>
</tr>
<tr>
<td>SegmentStatus</td>
<td>Flag indicating whether message retrieved is a segment of a logical message</td>
<td></td>
</tr>
<tr>
<td>Segmentation</td>
<td>Flag indicating whether further segmentation is allowed for the message retrieved</td>
<td></td>
</tr>
<tr>
<td>MsgToken</td>
<td>Message token</td>
<td></td>
</tr>
<tr>
<td>ReturnedLength</td>
<td>Length of message data returned (bytes)</td>
<td></td>
</tr>
<tr>
<td>MsgHandle</td>
<td>The handle to a message that is to be populated with the properties of the message being retrieved from the queue.</td>
<td></td>
</tr>
</tbody>
</table>

### Get Options

<table>
<thead>
<tr>
<th>Option Code</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQGMO-WAIT</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>MQGMO-WAIT</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>MQGMO-GET-SIGNAL</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>MQGMO-SYNCPNT</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>MQGMO-SYNCPNT</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>MQGMO-LOGICAL-ORDER</td>
<td></td>
<td>32768</td>
</tr>
<tr>
<td>MQGMO-COMPLETE-MQS</td>
<td></td>
<td>45528</td>
</tr>
<tr>
<td>MQGMO-ALL-MQS-AVAILABLE</td>
<td></td>
<td>131072</td>
</tr>
<tr>
<td>MQGMO-ALL-SEGMENTS-AVAILABLE</td>
<td></td>
<td>262144</td>
</tr>
<tr>
<td>MQGMO-NO-SYNCPNT</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>MQGMO-NO-LOGICAL-ORDER</td>
<td></td>
<td>65536</td>
</tr>
<tr>
<td>MQGMO-NO-COMPLETE-MQS</td>
<td></td>
<td>131072</td>
</tr>
<tr>
<td>MQGMO-NO-ALL-MQS-AVAILABLE</td>
<td></td>
<td>262144</td>
</tr>
<tr>
<td>MQGMO-NO-ALL-SEGMENTS-AVAILABLE</td>
<td></td>
<td>524288</td>
</tr>
<tr>
<td>MQGMO-MARK-BROWSE-HANDLE</td>
<td></td>
<td>1048576</td>
</tr>
<tr>
<td>MQGMO-MARK-BROWSE-CO-OP</td>
<td></td>
<td>2097152</td>
</tr>
<tr>
<td>MQGMO-UNMARK-BROWSE-CO-OP</td>
<td></td>
<td>4194304</td>
</tr>
<tr>
<td>MQGMO-UNMARK-BROWSE-HANDLE</td>
<td></td>
<td>8388608</td>
</tr>
<tr>
<td>MQGMO-UNMARKED-BROWSE-MQS</td>
<td></td>
<td>16777216</td>
</tr>
<tr>
<td>MQGMO-PROPERTIES-FORCE-MQRFH2</td>
<td></td>
<td>33554432</td>
</tr>
<tr>
<td>MQGMO-NO-PROPERTIES</td>
<td></td>
<td>67108864</td>
</tr>
<tr>
<td>MQGMO-PROPERTIES-IN-HANDLE</td>
<td></td>
<td>134217728</td>
</tr>
<tr>
<td>MQGMO-PROPERTIES-COMPATIBILITY</td>
<td></td>
<td>268435456</td>
</tr>
<tr>
<td>MQGMO-PROPERTIES-AS-Q-DEF</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Options can be added together as required.
MQGET Tips

- **Avoid using default syncpoint setting**
  - Defaults are not the same on z/OS and Distributed
  - Generally
    - MQGMO_SYNCPOINT_IF_PERSISTENT

- **Use MQGMO_FAIL_IF QUIESCING**
  - Ensure your application ends promptly

- **Generally use MQGMO_CONVERT**
  - Even if you ‘think’ you don’t need it

- **Remember to reset MsgId & Correlld fields**
  - These fields are used for selection and are returned

- **Handle ‘poison message’**
  - Look at BackoutCount in MQMD

- **Consider using MQCB to consume messages instead**
  - Callback semantics, often easier to code

---

Publish a message

- MQCONNX
- MQOPEN
- MQPUT

**Connection Handle**
- **Object Handle**
- **Message Descriptor**
- **Put Message Options**
- **Message Data**

**Completion Code**
- **Reason Code**

- Updates structure
  - Message Descriptor
  - Put Message Options

- Very similar to a normal P2P Put

---

QMGR
Publishing Application

- **MQOPEN** a topic
- **MQOD** describes a topic to publish to
  - MQOD-OBJECTTYPE
    - MQOT-Q for point-to-point
    - MQOT-TOPIC for publish
  - MQOD-OBJECTSTRING
  - MQOD-OBJECTNAME
- **MQPUT** a message

```
01 OBJECT-DESCRIPTOR.
COPY CMQODV.
MOVE MQOT-TOPIC TO MQOD-OBJECTTYPE.
MOVE MQOD-VERSION-4 TO MQOD-VERSION.
MOVE 'Price/Fruit/Apples' TO TARGET-TOPIC.
SET MQOD-OBJECTSTRING-VSPTR TO ADDRESS OF TARGET-TOPIC.
MOVE 18 TO MQOD-OBJECTSTRING-VSLENGTH.
```

Publishing Tips

- **Choose topic string carefully**
  - Use sensible topic hierarchy
    - Based on context of published data
  - Don’t use different topic for each publish
    - This is probably meta data, use message property
  - Topic strings can be up to 10K bytes
    - But don’t use long topics unless necessary

- **Consider using Topic object and Topic string**
  - Administer can set point in topic tree
    - Known as ‘topic tree isolation’
Subscribe to a topic

Connection Handle
Subscription Descriptor
Object Handle

Subscript Handle
Completion Code
Reason Code

Updates structure
- Subscription Descriptor
- Very similar to MQOPEN

Subscribing Application

- MQSUB verb
- Subscription Descriptor (MQSD) describes the topic
  - MQSD-OBJECTSTRING
  - MQSD-OBJECTNAME
- Consume publications from the returned Q-HANDLE
  - when MQSO-MANAGED used

CALL 'MQSUB'
USING HCONN,
  SUB-DESCRIPTOR,
  Q-HANDLE,
  SUB-HANDLE,
  MQCC,
  REASON.

CALL 'MQGET'
USING HCONN,
  Q-HANDLE,
  MESSAGE-DESCRIPTOR,
  GMOPTIONS,
  BUFFER-LENGTH,
  BUFFER,
  DATA-LENGTH,
  MQCC,
  REASON.

01 SUB-DESCRIPTOR.
  COPY CMQSDV.
  ADD MQSO-CREATE MQSO-MANAGED MQSO-FAIL-IF-QUIESCING
  GIVING MQSD-OPTIONS.
  MOVE 'Price/Fruit/Apples' TO TARGET-TOPIC.
  SET MQSD-OBJECTSTRING-VSPTR TO ADDRESS OF TARGET-TOPIC.
  MOVE 18 TO MQSD-OBJECTSTRING-VSLENGTH.
Subscription Descriptor (MQSD)

Field | Description
--- | ---
StrucId | Structure identifier
Version | Structure version number
Options | Options that control the action of MQSUB
ObjectName | Object Name
AlternateUserId | Alternate User Id
AlternateSecurityId | Alternate Security Id
SubExpiry | Subscription expiry
ObjectString | Object string
SubName | Subscription name
SubUserData | Subscription user data
PubPriority | Publication priority
PubAccountingToken | Publication accounting token
PubApplicationIdentityData | Publication application identity data
SelectionString | String providing selection criteria
SubLevel | Subscription Level
ResObjectString | Resolved object string

Subscribe Options

10 MQSO-NON-DURABLE PIC 9(9) BINARY VALUE 0.
10 MQSO-READ-AHEAD-AS-Q-DEF PIC 9(9) BINARY VALUE 0.
10 MQSO-ALTER PIC 9(9) BINARY VALUE 1.
10 MQSO-CREATE PIC 9(9) BINARY VALUE 2.
10 MQSO-RESUME PIC 9(9) BINARY VALUE 4.
10 MQSO-DURABLE PIC 9(9) BINARY VALUE 8.
10 MQSO-GROUP-SUB PIC 9(9) BINARY VALUE 16.
10 MQSO-MANAGED PIC 9(9) BINARY VALUE 32.
10 MQSO-FIXED-USERID PIC 9(9) BINARY VALUE 256.
10 MQSO-ANY-USERID PIC 9(9) BINARY VALUE 512.
10 MQSO-PUBLICATIONS-ON-REQUEST PIC 9(9) BINARY VALUE 2048.
10 MQSO-NEW-PUBLICATIONS-ONLY PIC 9(9) BINARY VALUE 4096.
10 MQSO-FAIL-IF-QUIESCING PIC 9(9) BINARY VALUE 8192.
10 MQSO-ALTERNATE-USER-AUTHORITY PIC 9(9) BINARY VALUE 262144.
10 MQSO-WILDCARD-CHAR PIC 9(9) BINARY VALUE 1048576.
10 MQSO-WILDCARD-TOPIC PIC 9(9) BINARY VALUE 2097152.
10 MQSO-SET-CORREL-ID PIC 9(9) BINARY VALUE 4194304.
10 MQSO-Scope-Queue PIC 9(9) BINARY VALUE 67108864.
10 MQSO-NO-READ-AHEAD PIC 9(9) BINARY VALUE 134217728.

Options can be added together as required
Subscribing Tips

- Managed handles make things simpler

- Only use durable subscriptions when necessary
  - Avoid build up of messages

- For durable subscriptions
  - Combine the create and resume options to make it simpler

Close a handle

- Connection Handle
- Object Handle Close Options
- Completion Code
- Reason Code
- Updates Object Handle
Closing Application

- **MQOPEN a queue**
- **MQCLOSE a queue**
  - Normally we’d do something!

## Close Options

Options available depending on object type

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCO-DELETE</td>
<td>1</td>
<td>Permanent Dynamic Queue</td>
</tr>
<tr>
<td>MQCO-DELETE-PURGE</td>
<td>2</td>
<td>Permanent Dynamic Queue</td>
</tr>
<tr>
<td>MQCO-KEEP-SUB</td>
<td>4</td>
<td>Durable Subscription</td>
</tr>
<tr>
<td>MQCO-REMOVE-SUB</td>
<td>8</td>
<td>Durable Subscription</td>
</tr>
<tr>
<td>MQCO-SUSPEND</td>
<td>32</td>
<td>Read Ahead input handle</td>
</tr>
</tbody>
</table>
**MQCLOSE Tips**

- **In triggered applications**
  - Only close triggered queue if application ending

- **If implementing queue cache**
  - Close ‘rarely used’ queues in a timely fashion
    - Open queues can not be deleted/purged and use memory

- **For read ahead queues**
  - Use the quiesce close option to avoid message loss

---

**Disconnect from Queue Manager**

- MQCONNX
- MQOPEN
- MQPUT
- MQOPEN
- MQGET
- MQCLOSE
- MQDISC

- **Connection Handle**
- **Completion Code**
- **Reason Code**

- Updates connection handle
Disconnecting Application

- **MQCONN to Queue Manager**
- **MQDISC from Queue Manager**
  - Normally we’d do something!

```
CALL 'MQCONN'
USING QM-NAME,
CONNECT-OPTIONS,
HCONN,
MQCC,
REASON.

< Issue some MQI calls here >

CALL 'MQDISC'
USING HCONN,
MQCC,
REASON.
```

MQDISC Tips

- **Ensure application disconnects if QM quiescing**
  - Will prevent Queue Manager from ending

- **MQDISC will close all queues/topics and subscriptions**
  - May wish to close some queues individually

- **MQDISC is an implicit commit**
  - May want to consider issuing MQBACK() first

- **Still call MQDISC**
  - If MQI call returns with a connection broken reason code

- **Application ending without MQDISC**
  - Will backout on Distributed
  - Will commit or backout depending on exit reason on z/OS
  - Try to always do explicit MQDISC if possible

```
05 CONNECTION-OPTIONS.
COPY CMQCNOV.

ADD MQCNO-HANDLE-SHARE-BLOCK MQCNO-RECONNECT
GIVING CONNECT-OPTIONS.
```
Summary

- **Simple MQI – very easy to get started**
  - Let most fields have default values
  - Keep things simple if you can, for example
    - do not try and monitor channels
    - do not try to inquire queue depths

- **Plenty of samples to help you along**
  - In a variety of languages
    - eg. `<install dir>\Tools\cobol\Samples`
    - `<hlq>.SCSQCOBS`

- **Check reason codes and log failures**
  - MQ trace can be useful

- **Also check out**
  - “An Introduction to and Comparison of the Different MQ APIs” by Matt Whitehead