Introduction to the MQI

Morag Hughson – morag@mqgem.com
MQGem Software

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

Agenda

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: 0
    - MQCC_WARNING: 1
    - MQCC_FAILED: 2
  - Reason Code
    - MQRC_????????: 2xxx
    - MQRC_NONE: 0

- **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 structure initialisers
  - MQMD md = { MQMD_DEFAULT };  Initialise to version 1

- Structures are versioned
  - Set the minimum version you need
    - md.Version = 2;
  - Don't use current version
    - md.Version = MQMD_CURRENT_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

- Basic connect
Connect with extended options

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

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

Connecting

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

- Connections options
  - Connection not thread specific
  - Client reconnect

```c
MQCONNX( Qm, &cno, &hQm, &CompCode, &Reason);

if (CompCode == MQCC_FAILED) {
    /* Do some error processing */
    /* Possibly retry */
}
```

```c
MQHCONN  hQm = MQHC_UNUSABLE_HCONN;
MQCHAR48 Qm  = "QM1";
MQCNO    cno = {MQCNO_DEFAULT};

cno.Options |= MQCNO_HANDLE_SHARE_BLOCK |
               MQCNO_RECONNECT;
```
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

MQCONN
MQOPEN

Connection Handle
Open 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

QMGR
Open a queue

- **MQOPEN a queue**
  - **OpenOptions**
    - MQOO_* flags which are required

- **MQOD describes a object to open**
  - **ObjectType**
    - MQOT_Q for point-to-point
    - MQOT_TOPIC for publish
  - ObjectString/ObjectName

```c
MQHOBJ hObj = MQHO_UNUSABLE_HOBJ;
MQOD ObjDesc = {MQOD_DEFAULT};

ObjDesc.ObjectType = MQOT_Q;
strcpy(ObjectDesc.ObjectName, "Q1");
```

---

**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>StructureVersion</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></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>Offset</td>
<td>Offset of first object record from start of MQOD</td>
<td>2</td>
</tr>
<tr>
<td>ResponseRecOffset</td>
<td>Offset of first response record from start of MQOD</td>
<td></td>
</tr>
<tr>
<td>ObjRecPtr</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>3</td>
</tr>
<tr>
<td>ResolvedQName</td>
<td>Resolved queue name</td>
<td></td>
</tr>
<tr>
<td>ResolvedQMgrName</td>
<td>Resolved queue manager name</td>
<td></td>
</tr>
<tr>
<td>ResolvedType</td>
<td>Resolved object type</td>
<td>4</td>
</tr>
<tr>
<td>ResolvedObjName</td>
<td>Resolved long object name</td>
<td></td>
</tr>
<tr>
<td>ResolvedString</td>
<td>Resolved object string</td>
<td></td>
</tr>
<tr>
<td>SelectionString</td>
<td>Selection string</td>
<td></td>
</tr>
<tr>
<td>ResolvedObject</td>
<td>Resolved object</td>
<td></td>
</tr>
</tbody>
</table>
Open Options

- #define MQOO_BIND_AS_Q_DEF          0x00000000
- #define MQOO_READ_AHEAD_AS_Q_DEF    0x00000000
- #define MQOO_INPUT_AS_Q_DEF         0x00000001
- #define MQOO_INPUT_SHARED           0x00000002
- #define MQOO_INPUT_EXCLUSIVE        0x00000004
- #define MQOO_BROWSE                  0x00000008
- #define MQOO_OUTPUT                  0x00000010
- #define MQOO_INQUIRE                 0x00000020
- #define MQOO_SET                      0x00000040
- #define MQOO_SAVE_ALL_CONTEXT       0x00000080
- #define MQOO_PASS_IDENTITY_CONTEXT  0x00000100
- #define MQOO_PASS_ALL_CONTEXT        0x00000200
- #define MQOO_SET_IDENTITY_CONTEXT    0x00000400
- #define MQOO_SET_ALL_CONTEXT         0x00000800
- #define MQOO_ALTERNATE_USER_AUTHORITY 0x00001000
- #define MQOO_FAIL_IF_QUIESCING       0x00002000
- #define MQOO_BIND_ON_OPEN            0x00004000
- #define MQOO_BIND_NOT_FIXED           0x00008000
- #define MQOO_CO_OP                     0x00020000
- #define MQOO_NO_READ_AHEAD            0x00080000
- #define MQOO_READ_AHEAD               0x00100000

Options can be ‘ORed’ together as required.

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 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

```c
MQMD md = {MQMD_DEFAULT};
MQPMO pmo = {MQPMO_DEFAULT};
char Msg = "Hello World!";

memcpy(md.Format, MQFMT_STRING, MQ_FORMAT_LENGTH);
pmo.Options = MQPMO_NO_SYNCPOINT;
```
## 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>Emty</td>
<td>Message defines</td>
</tr>
<tr>
<td>Feedback</td>
<td>Feedback or reason code</td>
</tr>
<tr>
<td>Encoding</td>
<td>Numeric encoding of message data</td>
</tr>
<tr>
<td>CodedCharSetId</td>
<td>Character set identifier of message data</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>Persistence</td>
<td>Message persistence</td>
</tr>
<tr>
<td>MsgId</td>
<td>Message identifier</td>
</tr>
<tr>
<td>CorrelId</td>
<td>Correlation identifier</td>
</tr>
<tr>
<td>BackoutCount</td>
<td>Backout counter</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>

<table>
<thead>
<tr>
<th>Field (V2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Group identifier</td>
</tr>
<tr>
<td>MsgSequencer</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>MsgFlags</td>
<td>Message flags</td>
</tr>
<tr>
<td>OriginalLength</td>
<td>Length of original message</td>
</tr>
</tbody>
</table>

## Put Message Options (MQPMO)

<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>1</td>
</tr>
<tr>
<td>Options</td>
<td>Options that control the action of MQPUT and MQPUT1</td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td>Object handle of input queue</td>
<td>1</td>
</tr>
<tr>
<td>KnownDestCount</td>
<td>Number of messages sent successfully to local queue</td>
<td></td>
</tr>
<tr>
<td>UnknownDestCount</td>
<td>Number of messages sent successfully to remote queue</td>
<td></td>
</tr>
<tr>
<td>InvalidDestCount</td>
<td>Number of messages that could not be sent</td>
<td></td>
</tr>
<tr>
<td>ResolvedQName</td>
<td>Resolved name of destination queue</td>
<td></td>
</tr>
<tr>
<td>ResolvedQMgrName</td>
<td>Resolved name of destination queue manager</td>
<td></td>
</tr>
<tr>
<td>RecsPresent</td>
<td>Number of put messages records or response records present</td>
<td>2</td>
</tr>
<tr>
<td>PutMsgRecFields</td>
<td>Flags indicating which MQPMR fields are present</td>
<td></td>
</tr>
<tr>
<td>PutMsgRecOffset</td>
<td>Offset of first put message record</td>
<td>2</td>
</tr>
<tr>
<td>ResponseRecOffset</td>
<td>Offset of first response record from start of MQPMO</td>
<td></td>
</tr>
<tr>
<td>ResponseRecPr</td>
<td>Address of first response record</td>
<td></td>
</tr>
<tr>
<td>OriginalMsgHandle</td>
<td>Original message handle</td>
<td></td>
</tr>
<tr>
<td>NewMsgHandle</td>
<td>New message handle</td>
<td>3</td>
</tr>
<tr>
<td>Action</td>
<td>Type of put being performed and the relationship between the original message and the new message</td>
<td></td>
</tr>
<tr>
<td>PubLevel</td>
<td>Level of subscription targeted by the publication</td>
<td></td>
</tr>
</tbody>
</table>
### Put Options

Options can be ‘ORed’ together as required

```c
#define MQPMO_SYNCPOINT                0x00000002
#define MQPMO_NO_SYNCPOINT             0x00000004
#define MQPMO_DEFAULT_CONTEXT          0x00000020
#define MQPMO_NEW_MSG_ID               0x00000040
#define MQPMO_NEW_CORREL_ID            0x00000080
#define MQPMO_PASS_IDENTITY_CONTEXT    0x00000100
#define MQPMO_PASS_ALL_CONTEXT         0x00000200
#define MQPMO_SET_IDENTITY_CONTEXT     0x00000400
#define MQPMO_SET_ALL_CONTEXT          0x00000800
#define MQPMO_ALTERNATE_USER_AUTHORITY 0x00001000
#define MQPMO_FAIL_IF_QUIESCING        0x00002000
#define MQPMO_NO_CONTEXT               0x00004000
#define MQPMO_LOGICAL_ORDER            0x00008000
#define MQPMO_ASYNC_RESPONSE           0x00010000
#define MQPMO_SYNC_RESPONSE            0x00020000
#define MQPMO_RESOLVE_LOCAL_Q          0x00040000
#define MQPMO_WARN_IF_NO_SUBS_MATCHED  0x00080000
#define MQPMO_RETAIN                   0x00200000
#define MQPMO_MD_FOR_OUTPUT_ONLY       0x00800000
#define MQPMO_SCOPE_QMGR               0x04000000
#define MQPMO_SUPPRESS_REPLYTO         0x08000000
#define MQPMO_NOT_OWN_SUBS             0x10000000
#define MQPMO_RESPONSE_AS_Q_DEF        0x00000000
#define MQPMO_RESPONSE_AS_TOPIC_DEF    0x00000000
```

### 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

```c
MQMD md = {MQMD_DEFAULT};
MQGMO gmo = {MQGMO_DEFAULT};
gmo.Options = MQGMO_SYNCPOINT_IF_PERSISTENT |
MQGMO_CONVERT |
MQGMO_WAIT |
MQGMO_FAIL_IF_QUIESCING;
gmo.WaitInterval = 60 * 1000;
```
### 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>1</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>ResolvedQName</td>
<td>Resolved 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

```c
#define MQGMO_WAIT             0x00000001
#define MQGMO_NO_WAIT          0x00000000
#define MQGMO_SET_SIGNAL       0x00000008
#define MQGMO_FAIL_IF_QUIESCING 0x00002000
#define MQGMO_SYNCPOINT        0x00000002
#define MQGMO_SYNCPOINT_IF_PERSISTENT 0x00001000
#define MQGMO_NO_SYNCPOINT     0x00000004
#define MQGMO_MARK_SKIP_BACKOUT 0x00000080
#define MQGMO_BROWSE_FIRST     0x00000010
#define MQGMO_BROWSE_NEXT      0x00000020
#define MQGMO_BROWSE_MSG_UNDER_CURSOR 0x00000800
#define MQGMO_BROWSE_MSG_UNDER_CURSOR_PERSISTENT 0x00000100
#define MQGMO_BROWSE_MSG_UNDER_CURSOR_PERSISTENT_LONG 0x00000020
#define MQGMO_BROWSE_MSG_UNDER_CURSOR_PERSISTENT_SHORT 0x00000010
#define MQGMO_UNLOCK           0x00000400
#define MQGMO_ACCEPT_TRUNCATED_MSG 0x00000040
#define MQGMO_CONVERT         0x00004000
#define MQGMO_LOGICAL_ORDER   0x00000000
#define MQGMO_COMPLETE_MSG    0x00000000
#define MQGMO_UNLOCK          0x00000400
#define MQGMO_ACCEPT_TRUNCATED_MSG 0x00000040
#define MQGMO_CONVERT         0x00004000
#define MQGMO_ALL_MSGS_AVAILABLE 0x00000200
#define MQGMO_ALL_SEGMENTS_AVAILABLE 0x00000400
#define MQGMO_MARK_BROWSE_HANDLE 0x00000100
#define MQGMO_MARK_BROWSE_CO_OP 0x00000200
#define MQGMO_UNMARK_BROWSE_CO_OP 0x00000400
#define MQGMO_UNMARK_BROWSE_HANDLE 0x00000800
#define MQGMO_PROPERTIES_FORCE_MQRFH2 0x00000100
#define MQGMO_PROPERTIES_IN_HANDLE 0x00000200
#define MQGMO_PROPERTIES_COMPATIBILITY 0x00000400
#define MQGMO_PROPERTIES_AS_Q_DEF 0x00000800
```
**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_IFQUIESCING**
  - Ensure your application ends promptly

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

- **Remember to reset MsgId & CorrelId 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**
Publishing Application

- MQOPEN a topic
  - MQOD describes a topic to publish to
    - ObjectType
      - MQOT_Q for point-to-point
      - MQOT_TOPIC for publish
    - ObjectString/ObjectName
  - MQPUT a message

```c
MQOD ObjDesc = {MQOD_DEFAULT};

ObjDesc.ObjectType = MQOT_TOPIC;
ObjDesc.Version = MQOD_VERSION_4;
ObjDesc.ObjectString.VSPtr = "Price/Fruit/Apples";
ObjDesc.ObjectString.VSLength = MQVS_NULL_TERMINATED;
```

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

- **MQCONN**
- **MQSUB**
- **MQGET**

QMGR

- Connection Handle
- Subscription Descriptor
- Object Handle
- Subscription 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 hObj
  - when MQSO_MANAGED used

MQSD
SubDesc = {MQSD_DEFAULT};
SubDesc.ObjectString.VSPtr = "Price/Fruit/Apples";
SubDesc.ObjectString.VSLength = MQVS_NULL_TERMINATED;
SubDesc.Options = MQSO_CREATE |
  MQSO_MANAGED |
  MQSO_FAIL_IF_QUIESCING;
### Subscription Descriptor (MQSD)

<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 action of MQSUB</td>
</tr>
<tr>
<td>ObjectName</td>
<td>Object Name</td>
</tr>
<tr>
<td>AlternateUserid</td>
<td>Alternate User Id</td>
</tr>
<tr>
<td>AlternateSecurityId</td>
<td>Alternate Security Id</td>
</tr>
<tr>
<td>SubExpiry</td>
<td>Subscription expiry</td>
</tr>
<tr>
<td>ObjectString</td>
<td>Object string</td>
</tr>
<tr>
<td>SubName</td>
<td>Subscription name</td>
</tr>
<tr>
<td>SubUserData</td>
<td>Subscription user data</td>
</tr>
<tr>
<td>PubPriority</td>
<td>Publication priority</td>
</tr>
<tr>
<td>PubAccountingToken</td>
<td>Publication accounting token</td>
</tr>
<tr>
<td>PubAppIdentityData</td>
<td>Publication application identity data</td>
</tr>
<tr>
<td>SelectionString</td>
<td>String providing selection criteria</td>
</tr>
<tr>
<td>SubLevel</td>
<td>Subscription Level</td>
</tr>
<tr>
<td>ResObjectString</td>
<td>Resolved object string</td>
</tr>
</tbody>
</table>

Options can be ‘ORed’ together as required

```c
#define MQSO_NON_DURABLE    0x00000000
#define MQSO_READ_AHEAD_AS_Q_DEF 0x00000000
#define MQSO_ALTER          0x00000001
#define MQSO_CREATE         0x00000002
#define MQSO_RESUME         0x00000004
#define MQSO_DURABLE        0x00000008
#define MQSO_GROUP_SUB      0x00000010
#define MQSO_MANAGE         0x00000020
#define MQSO_SET_IDENTITY_CONTEXT 0x00000040
#define MQSO_FIXED_USERID   0x00000080
#define MQSO_ANY_USERID     0x00000100
#define MQSO_PUBLICATIONS_ON_REQUEST 0x00000200
#define MQSO_NEW_PUBLICATIONS_ONLY 0x00000400
#define MQSO_FAIL_IF_QUIESCING 0x00000800
#define MQSO_ALTERNATE_USER_AUTHORITY 0x00001000
#define MQSO_WILDCARD_CHAR  0x00002000
#define MQSO_WILDCARD_TOPIC 0x00004000
#define MQSO_SET_CORREL_ID  0x00008000
#define MQSO_SCOPE_QMGR     0x00010000
#define MQSO_NO_READ_AHEAD  0x00020000
```
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!
  - Note address of MQHOBJ

```
MQHCONN hConn;
MQHOBJ hObj = MQHO_UNUSABLE_HOBJ;
MQOD ObjDesc = {MQOD_DEFAULT};

ObjDesc.ObjectType = MQOT_Q;
strcpy(ObjectDesc.ObjectName, "Q1");
```

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>0x00000001</td>
<td>Permanent Dynamic Queue</td>
</tr>
<tr>
<td>MQCO_DELETE_PURGE</td>
<td>0x00000002</td>
<td>Permanent Dynamic Queue</td>
</tr>
<tr>
<td>MQCO_KEEP_SUB</td>
<td>0x00000004</td>
<td>Durable Subscription</td>
</tr>
<tr>
<td>MQCO_REMOVE_SUB</td>
<td>0x00000008</td>
<td>Durable Subscription</td>
</tr>
<tr>
<td>MQCOQUIESCE</td>
<td>0x00000020</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 cannot 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!
  - Note address of MQHCONN

```c
MQHCONN  hConn = MQHC_UNUSABLE_HCONN;
MQCHAR48 Qm    = "QM1";
MQCNO    cno   = {MQCNO_DEFAULT};

cno.Options |= MQCNO_HANDLE_SHARE_BLOCK | MQCNO_RECONNECT
```

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
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\c\Samples`
    - `<hlq>.SCSQC37S`

- 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