MQI in C++

Code Sample
Connecting to the queue manager

To start working with MQSeries using this interface, you should first connect to a queue manager using one of the two available connection calls: MQCONN

MQCONN (QMgrName, Hconn, CompCode, Reason)

As input to MQCONN we must supply the name of the queue manager, or a null starting string if we want to open the default queue.

The output from these calls are:
- The result codes (completion and reason code).
- A connection handle to the queue manager.

The following code fragment shows how to connect to a queue manager using the C MQI:
MQCONN Hcon;     /* connection handle */
MQLONG CompCode;    /* completion code */
MQLONG Reason;      /* reason code */
char QMName[50];    /* queue manager name */
strcpy(QMName,"SampleQM");

// Connecting to the Queue Manager

MQCONN(QMName,    /* queue manager */
       &Hcon,       /* connection handle */
       &CompCode,   /* completion code */
       &CReason);  /* reason code */

if (CompCode == MQCC_FAILED) {
   printf("MQCONN failed with reason code %ld\n", CReason);
}
Opening MQSeries objects

Some types of MQSeries objects that can be opened:
- Queue
- Queue manager

To open any of these objects we use the MQOPEN call:

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

This call receives:
- A connection handle as returned by the MQCONN call.
- A description of the object we want to open, in the form of an object descriptor (MQOD) structure.
- One or more options that control the action of the call.

The output from this call is:
- The result codes (completion and reason code).
- An object handle that represents your access to the object.
- A modified object-descriptor structure, if the object opened was a dynamic queue.

For more information on how to open the other types of objects, please refer to the Application Programming Guide: http://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzal05.pdf

The following code fragment shows how to open a local or remote queue using the local queue manager name:
MQOD od = {MQOD_DEFAULT};        /* Object Descriptor */
MQLONG O_options;                /* MQOPEN options */

// Setting Queue Name in the Object Descriptor data structure

strcpy(od.ObjectName, "SampleQueue");

// Setting Open Options

O_options = MQOO_INPUT_AS_Q_DEF     /* open queue for input */
           + MQOO_FAIL_IF_QUIESCING; /* but not if MQM stopping */

// Opening the assigned object

MQOPEN(Hcon,                          /* connection handle */
       &od,                          /* object descriptor for queue */
       O_options,                   /* open options */
       &Hobj,                       /* object handle */
       &CompCode,                   /* completion code */
       &Reason);                    /* reason code */

// Here we use the reason code instead of the CompCode to show how
// this parameter can also be used to determine if any problem has occurred.

if (Reason != MQRC_NONE) {
    printf("MQOPEN failed with reason code %ld\n", Reason);
}
Putting messages in a queue

To put multiple messages in a queue we can use the MQPUT call:

MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode, Reason)

This call receives:
- A connection handle, as returned by the MQCONN call.
- A queue handle, as returned by the MQOPEN call.
- A description of the message you want to put on the queue, in the form of a message descriptor.
- Control information, in the form of a put-message options (MQPMO) structure.
- The length of the data contained in the message.
- The message itself.

The output of this call is:
- The result codes (completion and reason codes).
- Updated message descriptor and options if the call was executed successful.
/* Declare MQI structures needed */

MQOD od = {MQOD_DEFAULT};     /* Object Descriptor */
MQMD md = {MQMD_DEFAULT};     /* Message Descriptor */
MQPMO pmo = {MQPMO_DEFAULT};     /* put message options */
MQHCONN Hcon;         /* connection handle */
MQOBJ Hobj;         /* object handle */
MQLONG O_options; /* MQOPEN options */
MQLONG CompCode;       /* completion code */
MQLONG Reason;        /* reason code */
MQLONG messlen;        /* message length */
char buffer[100];        /* message buffer */

// A connection handle must be obtained using the MQCONN call
// as shown in the previous sections.
MQCONN(...);

// Setting the target queue name in the Object Descriptor data structure.
strncpy(od.ObjectName, "SampleQueue");

// Setting Open options, in this case MQOO_OUTPUT is require to execute
// the following MQPUT call.
O_options = MQOO_OUTPUT     /* open queue for output */
+ MQOO_FAIL_IF_QUIESCING;     /* but not if MQM stopping */
MQOPEN(Hcon,&od,O_options,&Hobj,&CompCode,&Reason);

// Validating the completion and reason codes
if (Reason != MQRC_NONE) {
    printf("MQOPEN ended with reason code %ld\n", Reason);
}
if (CompCode == MQCC_FAILED) {
    printf("unable to open queue for output\n");
}
Preparing message data. In this example a simple message data will be sent. Message data can be prepared using other resources such as files, information in a database, etc.

```c
strcpy(buffer, "Message data");
messlen = strlen(buffer);
```

The following two statements are not required if the `MQPMO_NEW_MSG_ID` and `MQPMO_NEW_CORREL_ID` options are used.

```c
memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
```

```c
MQPUT(Hcon, Hobj, &md, &pmo, messlen, buffer, &CompCode, &Reason);
/* report reason, if any */
if (Reason != MQRC_NONE) {
    printf("MQPUT ended with reason code %ld\n", Reason);
}
```

The queue must be closed after the last message has been sent, and we must disconnect from the queue manager.
To close an MQSeries object we use the MQCLOSE call.

MQCLOSE (Hconn, Hobj, Options, CompCode, Reason)

This call receive the following input:
- A connection handle.
- The handle of the object we want to close.
- The close options.

The output of this call is:
- The result codes (completion and reason code).
- The object handle, reset to the value MQHO_UNUSABLE_HOBJ.

Unless you are closing a permanent dynamic queue, the close options will be MQCO_NONE.

Typically a dynamic queue is deleted once the program that created it calls an MQCLOSE for that queue, but in the case of permanent dynamic queues, they can be retained by the queue manager or deleted depending on the options used in the MQCLOSE call.
MQLONG C_options;          /* MQCLOSE options */

C_options = 0;             /* no close options */
MQCLOSE(Hcon,              /* connection handle */
        &Hobj,              /* object handle */
        C_options,         /* completion code */
        &CompCode,         /* reason code */
        &Reason);
Getting messages from a queue

To get messages from a queue, we can use the MQGET call.

MQGET (Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer, DataLength, CompCode, Reason)

The input parameters for this call are:
- A connection handle.
- A queue handle.
- A description of the message we want to get from the queue in the form of an MQMD structure.
- Control information in the form of a Get Message Options (MQGMO) structure.
- The size of the buffer in which the message is going to be stored.
- The address of the buffer.

The output of this call is:
- The result codes (reason and completion codes).
- The message in the buffer specified, if the call completes successfully.
- The options structure, modified to show the name of the queue from which the message was retrieved.
- The message descriptor structure, with the information of the message that is retrieved.
- The actual length of the message.
MQMD md = MQMD_DEFAULT; /* Message Descriptor */
MQGMO gmo = MQGMO_DEFAULT; /* get message options */
MQHCONN Hcon; /* connection handle */
MQHOBJ Hobj; /* object handle */
MQLONG O_options; /* MQOPEN options */
MQLONG CompCode; /* completion code */
MQLONG Reason; /* reason code */
MQBYTE buffer[101]; /* message buffer */
MQLONG buflen; /* buffer length */
MQLONG messlen; /* message length received */
// A connection handle must be obtained using the MQCONN call
// as shown in the previous sections.
MQCONN(...);
// Setting the target queue name in the Object Descriptor data structure.
strncpy(od.ObjectName, "SampleQueue");
// Setting Open options, in this case MQOO_INPUT_AS_Q_DEF, MQOO_INPUT_SHARE
// or MQOO_INPUT_EXCLUSIVE is required to execute the following MQGET call.
O_options = MQOO_INPUT_AS_Q_DEF /* open queue for input */
+ MQOO_FAIL_IF_QUIESCING; /* but not if MQM stopping */
MQOPEN(Hcon, &od, O_options, &Hobj, &CompCode, &Reason);
// Validating the completion and reason codes
if (Reason != MQRC_NONE) {
    printf("MQOPEN ended with reason code %ld\n", Reason);
}
if (CompCode == MQCC_FAILED) {
    printf("unable to open queue for input\n");
}
// These options cause the MsgId and CorrelId to be replaced, so
// that there is no need to reset them before each MQGET if more than one
// get operation will be performed using the same get message options data
// structure. These options may not be available in all the environments.
gmo.Version = MQGMO_VERSION_2; /* Set Message Options version 2*/
// No match options will be used while scanning the queue
gmo.MatchOptions = MQMO_NONE;
gmo.Options = MQGMO_WAIT /* wait for new messages */
+ MQGMO_CONVERT; /* convert if necessary */
gmo.WaitInterval = MQWI_UNLIMITED; /* unlimited time for waiting */
buflen = sizeof(buffer) - 1; /* buffer size available for GET */
// Setting Encoding and CodedCharSetId in case some conversion is needed.
md.Encoding = MQENC_NATIVE;
md.CodedCharSetId = MQCCSI_Q_MGR;
// Perform the get operation
MQGET(Hcon,Hobj,&md,&gmo,buflen,buffer,&messlen,&CompCode,&Reason);
// report reason, if the call fails.
if (CompCode == MQCC_FAILED) {
printf("MQGET ended with reason code %ld\n", Reason);
} else {
// Display the message received
buffer[messlen] = '\0'; /* add terminator */
printf("message <\%s>\n", buffer);
}
// The queue must be closed after the last message has been received, and
// you must disconnect from the queue manager.