

## MSMQ-VC++ program-query

**Source:** <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2004-08/0501.html>

---

**From:** Thirumalai Internal (*thirumalai\_at\_cspl.com*)

**Date:** 08/13/04

Date: Fri, 13 Aug 2004 13:58:51 +0530

My query is about MSMQ(Microsoft Message Queuing) . I have written the codings in VC++ environment.

2 programs namely Sender Program and Receiver Program

The Sender program creates a Transactional Queue and sends messages to it

The Receiver side program accepts the messages from the Transactional Queue.

My Requirement is:

The sender side program(process) should send messages to the transactional queue and it should intimate the receiver side program(process),which is involved in some other work, that it has send some messages to it.

How the sender side program(process) can give intimation to the receiver side program that it has send messages for the receiver?

Here is the Sender side program:

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <mq.h>

int main(void)
{

    QUEUEPROPID aPropId[10];
    MQPROPVARIANT aVariant[10];
    HRESULT hr;
    HRESULT aStatus[10];
    MQQUEUEPROPS QueueProps;
    DWORD PropIdCount=0;

    int counter=0;
    int times=1000;
```

```
LPWSTR wszPathName = L"cspl\\myPublicQueue76"; //Queue Pathname
```

```
aPropId[PropIdCount] = PROPID_Q_PATHNAME;
aVariant[PropIdCount].vt = VT_LPWSTR;
aVariant[PropIdCount].pwszVal = wszPathName;
PropIdCount++;
```

```
aPropId[PropIdCount] = PROPID_Q_PATHNAME;
aVariant[PropIdCount].vt = VT_NULL;
PropIdCount++;
```

```
aPropId[PropIdCount] = PROPID_Q_TRANSACTION;
aVariant[PropIdCount].vt = VT_UI1;
aVariant[PropIdCount].bVal = MQ_TRANSACTIONAL;
PropIdCount++;
```

```
WCHAR wszLabel[MQ_MAX_Q_LABEL_LEN] = L"Test Message";
aPropId[PropIdCount] = PROPID_Q_LABEL;
aVariant[PropIdCount].vt = VT_LPWSTR;
aVariant[PropIdCount].pwszVal = wszLabel;
PropIdCount++;
```

```
QueueProps.cProp = PropIdCount;
QueueProps.aPropID = aPropId;
QueueProps.aPropVar = aVariant;
QueueProps.aStatus = aStatus;
```

```
WCHAR szFormatNameBuffer[1024];
DWORD dwFormatNameBufferLength = 1024;
```

```
hr = MQCreateQueue( NULL, //Creating Queue
    &QueueProps,
    szFormatNameBuffer,
    &dwFormatNameBufferLength
);
```

```
if(hr==MQ_ERROR_QUEUE_EXISTS )
{
printf("ERROR:MQ_ERROR_QUEUE_EXISTS \n");
}
```

```
if (FAILED(hr))
{
printf("Failure of hr\n");
}
else
{
printf("Message queue creation is success\n");
}
```

```
//SEND THE MESSAGES TO THE DESTINATION QUEUE
```

```

const int NUMBEROFPROPERTIES = 5;
DWORD cPropId = 0;
HRESULT hr1;
HANDLE hQueue;
WCHAR * wszQueueName=L"myPublicQueue76";
WCHAR * wszComputerName=L"cspl";

WCHAR * wszFormatName = new WCHAR[wcslen(wszComputerName) +
wcslen(wszQueueName) + 12];

swprintf(wszFormatName,
    L"DIRECT=OS:%s\\%s",
    wszComputerName,
    wszQueueName
);

MQMSGPROPS msgProps;
MSGPROPID aMsgPropId[NUMBEROFPROPERTIES];
MQPROP VARIANT aMsgPropVar[NUMBEROFPROPERTIES];
HRESULT aMsgStatus[NUMBEROFPROPERTIES];

aMsgPropId[cPropId] = PROPID_M_LABEL;
aMsgPropVar[cPropId].vt = VT_LPWSTR;
aMsgPropVar[cPropId].pwszVal = L"Test Message";
cPropId++;

DWORD dwBodyType = VT_BSTR;
aMsgPropId[cPropId] = PROPID_M_BODY_TYPE;
aMsgPropVar[cPropId].vt = VT_UI4;
aMsgPropVar[cPropId].ulVal = dwBodyType;
cPropId++;

WCHAR wszMessageBody[] = L"Hai my name is thirumalairajan";
//Actual Message Body

aMsgPropId[cPropId] = PROPID_M_BODY;
aMsgPropVar[cPropId].vt = VT_VECTOR | VT_UI1 ;
aMsgPropVar[cPropId].caub.pElems = (LPBYTE)wszMessageBody;
aMsgPropVar[cPropId].caub.cElems = sizeof(wszMessageBody);
cPropId++;

printf("MessageBody=%s\t Size of
Body=%d\n",wszMessageBody,sizeof(wszMessageBody));

aMsgPropId[cPropId] = PROPID_M_DELIVERY;
aMsgPropVar[cPropId].vt = VT_UI1;
aMsgPropVar[cPropId].pwszVal = MQMSG_DELIVERY_EXPRESS ;
cPropId++;

aMsgPropId[cPropId] = PROPID_M_DEST_QUEUE;
aMsgPropVar[cPropId].vt = VT_LPWSTR;

```

```
aMsgPropVar[cPropId].pwszVal = wszFormatName;//szFormatNameBuffer;
cPropId++;
```

```
msgProps.cProp = cPropId;
msgProps.aPropID = aMsgPropId;
msgProps.aPropVar = aMsgPropVar;
msgProps.aStatus = aMsgStatus;
```

```
hr1 = MQOpenQueue(
    wszFormatName, //wszFormatName,
    MQ_SEND_ACCESS,
    MQ_DENY_NONE,
    &hQueue
);
```

```
if (FAILED(hr1))
{
    printf("Open Queue was a failure one\n");
    return hr1;
}
```

```
if(hr1==MQ_OK)
{
    printf("Open Queue was a success\n");
}
```

```
while(times>0)
{
```

```
    hr1 = MQSendMessage(
        hQueue,
        &msgProps,
        MQ_SINGLE_MESSAGE
    );
```

```
if (FAILED(hr1))
{

    printf("Send message was a failure one\n");
    return hr1;
}
else
{
    printf("Success in sending the message number %d",counter++);
    printf("\n");
}
times--;
Sleep(10);

}
```

```
// Close the Queue

hr1=MQCloseQueue(hQueue);

if (FAILED(hr1))
{
printf("\nFailed in closing the Queue\n");
return hr1;
}
else
{
printf("Success in closing the queue\n");
}

return hr;
}
```

Here is the Receiver side program:

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <mq.h>

int main(void)
{

HRESULT hr;
HANDLE hQueue;
WCHAR * wszQueueName=L"myPublicQueue76";
WCHAR * wszComputerName=L"cspl";

int counter=0;

int size;

//RECEIVE THE MESSAGES FROM THE DESTINATION QUEUE

const int NUMBEROFPROPERTIES = 5;
const int LABELSIZE = 250;
DWORD cPropId = 0;

MQMSGPROPS msgProps;
MSGPROPID aMsgPropId[NUMBEROFPROPERTIES];
MQPROPVARIANT aMsgPropVar[NUMBEROFPROPERTIES];
HRESULT aMsgStatus[NUMBEROFPROPERTIES];

aMsgPropId[cPropId] = PROPID_M_LABEL_LEN;
aMsgPropVar[cPropId].vt = VT_UI4;
aMsgPropVar[cPropId].ulVal = LABELSIZE;
cPropId++;
```

```

WCHAR wszLabelBuffer[LABELSIZE];
aMsgPropId[cPropId] = PROPID_M_LABEL;
aMsgPropVar[cPropId].vt = VT_LPWSTR;
aMsgPropVar[cPropId].pwszVal = wszLabelBuffer;
cPropId++;

aMsgPropId[cPropId] = PROPID_M_BODY_SIZE;
aMsgPropVar[cPropId].vt = VT_UI4;
cPropId++;

aMsgPropId[cPropId] = PROPID_M_BODY_TYPE;
aMsgPropVar[cPropId].vt = VT_UI4;
cPropId++;

char *buffer;

DWORD dwBodyBufferSize = 1024;
UCHAR *pucBodyBuffer = (UCHAR *)malloc(dwBodyBufferSize);
aMsgPropId[cPropId] = PROPID_M_BODY;
aMsgPropVar[cPropId].vt = VT_VECTOR|VT_UI1;
aMsgPropVar[cPropId].caub.pElems = (UCHAR *)pucBodyBuffer;
aMsgPropVar[cPropId].caub.cElems = dwBodyBufferSize;
cPropId++;

msgProps.cProp = cPropId;
msgProps.aPropID = aMsgPropId;
msgProps.aPropVar = aMsgPropVar;
msgProps.aStatus = aMsgStatus;

WCHAR * wszFormatName = new WCHAR[wcslen(wszQueueName) +
wcslen(wszComputerName) + 12];

swprintf(wszFormatName,
    L"DIRECT=OS:%s\\%s",
    wszComputerName,
    wszQueueName
);

hr = MQOpenQueue(
    wszFormatName,//wszFormatName,
    MQ_RECEIVE_ACCESS,
    MQ_DENY_NONE,
    &hQueue
);

if (FAILED(hr))
{
    printf("Open Queue was a failure one\n");
    return hr;
}

```

```

if(hr==MQ_OK)
{
printf("Open Queue was a success\n");
}

while (1)
{

    hr = MQReceiveMessage(hQueue,
        1000,
        MQ_ACTION_RECEIVE,
        &msgProps,
        NULL,
        NULL,
        NULL,
        MQ_NO_TRANSACTION
    );
    if (FAILED(hr))
    {
        wprintf(L"No messages. Closing queue\n");
        break;
// return hr;
    }
    else
    {
        printf("\nsuccess in receiving the message number %d",counter++ );
        printf("\n");
        //fwrite(wszMessageBody,1,size,stdout);
        printf("Message Body=%s\n",pucBodyBuffer);
    }

}

} //While ends here
// If the message contains a label, print it.
if (msgProps.aPropVar[0].ulVal == 0)
{
    wprintf(L"Removed message from queue\n");
}
else
{
    wprintf(L"Removed message '%s' from queue\n", wszLabelBuffer);
}
// }

// Close the queue.
hr = MQCloseQueue(hQueue);
if (FAILED(hr))
{
    return hr;
}

```

```
return hr;
```

```
}
```