

# Queue Management Services APIs

## Introduction to QMS

Queue Management Services (QMS) allow an application to create queues for controlling the flow of jobs and services on the network. A queue organizes client requests for a job server. A job server is software that resides at a specific workstation and provides services for other workstations on the network. Networks can have many different kinds of job servers, including print servers, archiving servers, compiling servers, message-sending servers and so on. By placing requests into network queues, a job server can provide service that is both flexible and efficient.

Some of the function calls in this chapter would only be called by a queue server. Others might be called by user applications which submit queue jobs and maintain created queues.

## NWAbortServicingQueueJob

This function signals the queue management software that a job cannot be completed successfully.

### Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
uint16       jobNumber;
NWFileHandle_ ta    fileHandle;

ccode=NWAbortServicingQueueJob( serverConnID, queueID, jobNumber, fileHandle );
```

### Input

*serverConnID* passes the job server connection ID.

*queueID* passes the bindery object ID for the queue in which the aborted job is located.

*jobNumber* passes the job number of the aborted job.

*fileHandle* passes a pointer to the file handle of the file associated with the aborted job.

### Output

None.

### Return Value

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job

0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

**Note:** Because `NWCloseFile` function is called with `NWAbortServicingQueueJob`, you may receive an `NWErrno` = `0x001100xx`. The `0x0011` indicates a file system services error. See Appendix B for a complete listing of possible NetWare errors.

## Description

This function call allows a job server to inform the queue manager that it cannot complete servicing a job previously accepted for service. This function closes the job file and resets the job server's access rights to their original (login) values.

An aborted job returns to its former position in the job queue if its Service Restart flag (bit 0x10 of the `jobControlFlags` field in the `NWQueueJobStruct_t`) is set. For example, if a job is at the beginning of the queue before being called, it returns to the beginning of the queue after being aborted. An aborted job could, therefore, be next in line for service. For this reason, a job should not be aborted because of an error in the job's format or requests. Instead, use the `NWFinishServicingQueueJob` function.

## Notes

A job should be aborted only if some temporary internal problem prevents it from completing. For example, a print job might be aborted if the printer has a paper jam. After the paper jam is corrected, the job server can service the job successfully.

**IMPORTANT:** If a job is attempting to access data without proper security clearance and is aborted, the job will remain in the queue and be serviced and aborted again and again. To remove a job from the job queue, a user would have to use the `NWCloseFileAndAbortQueueJob` call, or the queue server would have to use the `NWFinishServicingQueueJob` call.

Only a queue server that has previously accepted a job for service can make this function call.

## See Also

`NWChangeQueueJobEntry`  
`NWCreateQueueFile`  
`NWFinishServicingQueueJob`  
`NWReadQueueJobEntry`

## NWAttachQueueServerToQueue

This function attaches the calling client to the specified queue as a queue server.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16      serverConnID;
uint32      queueID;

ccode=NWAttachQueueServerToQueue( serverConnID, queueID );
```

**Input**

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue being attached.

**Output**

None.

**Return Value**

0	Successful.	
-1	Unsuccessful.	One of the following error codes is placed in NWErrno.
	0x99 (153)	Directory Full
	0xD0 (208)	Queue Error
	0xD1 (209)	No Queue
	0xD2 (210)	No Queue Server
	0xD3 (211)	No Queue Rights
	0xD4 (212)	Queue Full
	0xD5 (213)	No Queue Job
	0xD6 (214)	No Job Right
	0xD7 (215)	Queue Servicing
	0xD8 (216)	Queue Not Active
	0xD9 (217)	Station Not Server
	0xDA (218)	Queue Halted
	0xDB (219)	Max. Queue Servers
	0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

**Description**

This function call is created for queue job servers and must be used before the job server can perform any services in the queue. After the queue server has logged in to the file server as a queue server (bindery object), this call establishes a connection between the queue server and the queue. If the queue server logs out of the file server, this connection to the queue will be detached.

**Notes**

A client must attach itself to a queue as a job server before it can service jobs from that queue. A queue can have as many as 25 job servers attached. The workstation making this function call must be security equivalent to one of the objects listed in the queue's Q\_SERVERS group property.

**NWChangeQueueJobEntry**

This function changes the information about a job in a queue.

**Synopsis**

```
#include "nwapi.h";

int          ccode;
uint16       serverConnID;
uint32       queueID;
NWQueueJobStruct_t  jobStruct;

ccode=NWChangeQueueJobEntry( serverConnID, queueID, &jobStruct );
```

**Input**

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue.

*jobStruct* passes a pointer to the job structure that contains the new information about the job. (See Appendix A, <sup>a</sup>NWQueueJobStruct\_t Structure.)

**Output**

None.

**Return Value**

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno.
0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

**Description**

The following fields in the NWQueueJobStruct\_t structure may be changed by the owner of the job or by a queue operator:

- targetServerID
- targetExecutionTime
- jobType
- jobControlFlags
- jobDescription
- queueRecord

If the caller is an operator, the Operator Hold flag can be reset to a value supplied by the caller.

Use NWChangeQueueJobPosition to change the job's service position in the queue.

**Notes**

The NWChangeQueueJobEntry function can be used in conjunction with the NWReadQueueJobEntry function to change a portion of the job's entry record. However, if the target entry is already being serviced, the NWChangeQueueJobEntry function returns a servicing error and makes no changes to the job's entry record.

If this call is being used in conjunction with printing and the NWPrintStruct\_t, the structure must first be converted

(using the  
NWConvertPrintStructToQueueStruct) before this call is made.

See Also

NWChangeQueueJobEntry  
NWChangeQueueJobPosition  
NWConvertPrintStructToQueueStruct  
NWGetQueueJobList  
NWReadQueueJobEntry  
NWRemoveJobFromQueue

NWChangeQueueJobPosition

This function changes a job's position in a queue.

Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
uint16       jobNumber;
uint8        newJobPosition;

ccode=NWChangeQueueJobPosition( serverConnID, queueID, jobNumber,  newJobPosition );
```

Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the affected queue.

*jobNumber* passes the job number of the job being repositioned.

*newJobPosition* passes the job's new position.

Output

None.

Return Value

0        Successful.

-1       Unsuccessful.    One of the following error codes is placed in NWErrno.

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted

0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

The value of the newJobPosition parameter ranges from 1 to 250. Position 1 is the first position in the queue and position 250 is the last position in a full queue. If a specified position number places the job beyond the current end of the queue, the job is placed at the end of the current queue.

## Notes

When a job is moved in the queue, the positions of all job entries are updated to reflect the change. Changing the position of a job being serviced has no effect on the service of that job. Be aware that job positions change as other jobs in the queue are finished being serviced.

The application making this call must be logged in as supervisor.

## See Also

NWChangeQueueJobEntry  
 NWGetQueueJobList  
 NWReadQueueJobEntry  
 NWRemoveJobFromQueue

## NWChangeToClientRights

This function changes a queue server's current login identity to match the identity of the client for whom the queue server is acting.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16      serverConnID;
uint32      queueID;
uint16      jobNumber;

ccode=NWChangeToClientRights( serverConnID, queueID, jobNumber );
```

## Input

*serverConnID* passes the queue server connection ID.

*queueID* passes the bindery object ID of the queue.

*jobNumber* passes the job's job number.

## Output

None.

## Return Value

0      Successful.  
 -1     Unsuccessful. One of the following error codes is placed in NWErrno.

0x99 (153)	Directory Full
------------	----------------

0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows a queue server to change its current login identity to match the identity of the client for which it is acting. This is useful if the queue server must access files owned by the client but not submitted to the queue by the client (in other words, if the server must go out and retrieve files by itself). The queue server's login user ID and associated security equivalence list are replaced by the ID and security equivalence list of the user who placed the job in the queue.

This function does not change any path mappings that the queue server may have on the job server. However, all access rights to those directories are recalculated to conform to the rights of the queue client. Files opened before this call is made will continue to be accessible with the server's rights. Files opened after this call is made will be accessible only with the client's rights.

## Notes

The job server is responsible for creating any path mappings that it may need to carry out the client's requests after this call has been made.

The NWRestoreQueueServerRights function reverses the effects of the NWChangeToClientRights function. In addition, the server's rights are automatically reset if the server issues a NWFinishServicingQueueJob or NWAbortServicingQueueJob function.

Only a queue server that has previously accepted a job for service can call this function.

## See Also

NWAbortServicingQueueJob  
 NWFinishServicingQueueJob  
 NWRestoreQueueServerRights

## NWCloseFileAndAbortQueueJob

This function signals the QMS that a job has not been created properly and should be removed from the queue.

## Synopsis

```
#include "nwapi.h"
```

```
int          ccode;
uint16       serverConnID;
uint32       queueID;
uint16       jobNumber;
```

NWFileHandle\_t\* fileHandle;

ccode=NWCloseFileAndAbortQueueJob( serverConnID, queueID,  
jobNumber, fileHandle );

## Input

*serverConnID* passes the queue server connection ID.

*queueID* passes the bindery object ID of the affected queue.

*jobNumber* passes the job entry number of the job whose service is being aborted.

*fileHandle* passes a pointer to the file handle of the aborted job's file (returned from the NWCreateQueueFile function call).

## Output

None.

## Return Values

0 Successful.  
-1 Unsuccessful. One of the following error codes is placed in NWErrno.

0xD1	No Queue
0xD3	No Queue Rights
0xD4	Queue Full
0xD5	No Queue Job
0xF5	No Such Object
0xD6	No Job Right
0xD7	Queue Servicing
0xD8	Queue Not Active
0xFF	Invalid File Handle
0x30	Invalid Connection ID

**Note:** Because this API uses NWCloseFile, it is possible to get an NWErrno = 0x001100xx. The 0x0011 signifies a file system error. See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows the client to close a queue job and abort it. The jobNumber parameter contains the job number returned by QMS when the job was originally entered in the queue. The file associated with that job number is closed, and the job is deleted from the queue.

## Notes

Only the client that created the queue job can call this function.

## See Also

NWCloseFileAndStartQueueJob  
NWCreateQueueFile  
NWRemoveJobFromQueue



# NWCloseFileAndStartQueueJob

This function closes a queue file and marks it ready for execution.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
uint16       jobNumber;
NWFileHandle _ta    fileHandle;

ccode=NWCloseFileAndStartQueueJob( serverConnID, queueID,
jobNumber,  fileHandle );
```

## Input

*serverConnID* passes the queue server connection ID.

*queueID* passes the bindery object ID of the queue in which the specified job was placed.

*jobNumber* passes the job number of the job to be serviced.

*fileHandle* passes a pointer to the file handle of the file associated with the job to be executed (returned from the NWCreateQueueFile function call).

## Output

None.

## Return Values

- 0 Successful.
- 1 Unsuccessful. One of the following error codes is placed in NWErrno.
  - 0xD1 No Queue
  - 0xD3 No Queue Rights
  - 0xD5 No Queue Job
  - 0xF5 No Such Object
  - 0xD7 Queue Servicing
  - 0xD8 Queue Not Active
  - 0xFF Invalid File Handle
  - 0x30 Invalid Connection ID

**Note:** Because this API uses NWCloseFile, it is possible to get an NWErrno = 0x001100xx. The 0x0011 signifies a file system error. See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows the workstation to close a queue job file and mark the job for execution.

The jobNumber parameter contains the job number returned by QMS when the job was originally entered in the queue.

When this function finishes, the specified job is ready for execution, if the userHoldFlag and operatorHoldFlag fields are both cleared and the

targetExecutionTime was either not specified or has elapsed (set with the jobStruct parameter when the file was created).

Notes

Only the client that created the job can call this function.

See Also

NWCloseFileAndAbortQueueJob  
NWCreateQueueFile  
NWRemoveJobFromQueue

NWConvertPrintStructToQueueStruct

This function converts the printRecord to a form acceptable to NetWare print servers.

Synopsis

```
#include "nwapi.h";

int ccode;
NWPrintRecord_t printRecord;
NWClientRecord_ta queueRecord;

ccode=NWConvertPrintStructToQueueStruct( &printRecord, queueRecord );
```

Input

*printRecord* passes the address of the filled in printRecord structure. (See Appendix A, "NWPrintRecord\_t Structure.")

*queueRecord* passes a pointer to where all of the print information will be stored. (See Appendix A, the queueRecord field in "NWQueueJobStruct\_t Structure.")

Output

*queueRecord* receives the print record information. (See Appendix A, the queueRecord field in "NWQueueJobStruct\_t Structure.")

Description

Before this function can be called, all the fields in the NWPrintRecord\_t structure must be assigned values. The following sample code assumes that a NWPrintRecord\_t structure called prtR1 has been declared. (To automatically set all fields to 0, use memset or declare a static structure.)

versionNumber	Assigns the version number.
	<b>prtR1.versionNumber = 1;</b>
tabSize	Assigns the number of spaces tabs will be expanded to (0 -18). The following sample code expands the tab to 8 spaces:
	<b>prtR1.tabSize = 8;</b>

numCopies	<p>Assigns the number of copies that will be printed. The following sample code assigns 2 copies:</p> <pre><b>prtR1.numCopies = 2;</b></pre>
controlFlags	<p>Sets one or more control flags. Use one or more of the following:</p> <pre>NWPCF_SUPPRESS_FF NWPCF_NOTIFY_USER NWPCF_TEXT_MODE NWPCF_PRINT_BANNER</pre> <p>The following sample code suppresses the form feed:</p> <pre><b>prtR1.controlFlags =</b> NWPCF_SUPPRESS_FF;</pre>
linesPerPage	<p>Assigns the number of lines on one page. The following sample code assigns 66 lines per page:</p> <pre><b>prtR1.linesPerPage = 66;</b></pre>
charsPerLine	<p>Assigns the number of characters on one line. The following code assigns 132 characters per line:</p> <pre>prtR1.charsPerLine = 132;</pre>
formName	<p>Sets the form to use for printing the job. The following sample code sets this field to 0:</p> <pre>prtR1.formName[0] = 0;</pre>
bannerNameField	<p>Assigns the text that is printed in the first box in the banner. Usually the user's name is printed in this box. The following sample code sets this field to a user named Nikki:</p> <pre>strcpy(prtR1.bannerNameField, "Nikki");</pre>
bannerFileField	<p>Assigns the text that is printed in the second box in the banner. Usually the file name is printed in this box. The following sample code sets this field to a file name of API.DOC:</p> <pre>strcpy(prtR1.bannerFileField, "API.DOC");</pre>
headerFileName	<p>Assigns the file name that is printed in the header of the banner. The following sample code sets this field to 0:</p> <pre>prtR1.headerFileName[0] = 0;</pre>
directoryPath	<p>Assigns the full path name of the directory where the</p>

file resides. The following sample code assigns the file API.DOC in the SYS:DOC/API directory:

```
strcpy(prtR1.directoryPath,  
"SYS:DOC/API/API.DOC");
```

This function performs any necessary byte swapping and word alignment and then copies the printRecord into the queueRecord field in the NWQueueJobStruct\_t structure. This call is usually used before calling NWCreateQueueFile.

The queueRecord field contains information pertaining to the print job. This information is assigned by the client and is sometimes referred to as the "client record area."

## See Also

NWCreateQueueFile  
NWConvertQueueStructToPrintStruct

## NWConvertQueueStructToPrintStruct

This function converts the queueRecord (in the NWQueueJobStruct\_t) to a printRecord which is in the form originally passed in by the client.

## Synopsis

```
#include "nwapi.h";  
  
int                ccode;  
NWClientRecord_ta queueRecord;  
NWPrintRecord_t   printRecord;  
  
ccode=NWConvertQueueStructToPrintStruct( queueRecord,  &printRecord );
```

## Input

*queueRecord* passes the address of the print information. (See Appendix A, the queueRecord field in "NWQueueJobStruct\_t Structure.")

*printRecord* passes a pointer to the allocated NWPrintRecord\_t structure. (See Appendix A, "NWPrintRecord\_t Structure.")

## Output

*printRecord* fills in the converted queueRecord. (See Appendix A, "NWPrintRecord\_t Structure.")

## Description

This function performs any necessary byte swapping and word alignment and then copies the data back into the printRecord space. This call is necessary for the user to be able to read the fields in the print structure. This function is most often used after NWReadQueueJobEntry.

The queueRecord field in the NWQueueJobStruct\_t structure is an area which is filled in by the client; it is sometimes referred to as "client record area."

## See Also

NWReadQueueJobEntry  
NWConvertPrintStructToQueueStruct

# NWCreateQueue

This function creates a new queue in the bindery and file system of the specified file server.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
char         queueName[NWMAX_QUEUE_NAME_LENGTH];
uint16       queueObjectType;
NWDirHandle_ts directoryHandle;
char         queueSubdirectory[NWMAX_QUEUE
                                SUBDIR_LENGTH];

uint32       newQueueID;

ccode=NWCreateQueue( serverConnID, queueName, queueObjectType,
directoryHandle, queueSubdirectory, &newQueueID );
```

## Input

*serverConnID* passes the file server connection ID.

*queueName* passes a pointer to the name of queue to be created (48 characters).

*queueObjectType* passes a number indicating the bindery object type for the new queue.

*directoryHandle* passes the NetWare directory handle pointing to the directory in which the queue's property is to be created (0 if the queueSubdirectory parameter contains the full path).

*queueSubdirectory* passes a pointer to the absolute path or a path relative to the NetWare directory handle that will contain the queue files (119 characters, stored in the Q\_DIRECTORY property).

*newQueueID* passes a pointer to the space allocated for the new queue ID number.

## Output

*newQueueID* receives the new queue ID number.

## Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno.
0x99	Directory Full
0xFF	Failure
0xF5	No Object Create Privilege
0x30	Invalid Connection ID
0x9B	Invalid Dir Handle
0x98	Volume Does Not Exist
0xEE	Queue Exists

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function creates a queue in the bindery, using the type and name specified by the queueObjectType and

queueName parameters. Novell has the following bindery object types defined for queues:

```
NWOT_PRINT_QUEUE
NWOT_ARCHIVE_QUEUE
NWOT_JOB_QUEUE
```

This function also creates the Q\_DIRECTORY property. The value for the Q\_DIRECTORY property is determined by combining the directoryHandle and queueSubdirectory parameters.

QMS will use the directory handle and directory path parameters to create a queue directory that holds the system files containing the queue itself and the job files related to the queue entries. The directory path SYS:SYSTEM is commonly used for the queue directory. QMS uses this directory to store queue files until they are serviced.

Next, this function creates the following group properties:

- Q\_SERVERS
- Q\_OPERATORS
- Q\_USERS

## Notes

Only SUPERVISOR or a bindery object that is security equivalent to SUPERVISOR can create a queue.

## See Also

NWDestroyQueue

## NWCreateQueueFile

This function creates a queue file.

## Synopsis

```
#include "nwapi.h"
```

```
int                ccode;
uint16             serverConnID;
uint32             queueID;
NWQueueJobStruct_t jobStruct;
NWFileHandle_t     fileHandle;
```

```
ccode=NWCreateQueueFile( serverConnID, queueID, &jobStruct, fileHandle );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery's object ID for the queue.

*jobStruct* passes a pointer to the structure in which the information about the job is stored. (See Appendix A, "NWQueueJobStruct\_t Structure.")

*fileHandle* passes a pointer to the file handle of the file to be created in the queue.

## Output

*jobStruct* receives the completed job structure. (See Appendix A, "NWQueueJobStruct\_t Structure.")

*fileHandle* receives the file handle of the job's associated file. (This file contains data pertaining to the job; for example, a print job file would contain the actual data to be printed.)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno.
0x99	Directory Full
0xD7	Queue Servicing
0xD0	Queue Error
0xD8	Queue Not Active
0xD1	No Queue
0xDA	Queue Halted
0xD3	No Queue Rights
0xFC	No Such Object
0xD4	Queue Full
0xFF	Failure
0x30	Invalid Connection ID

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function allows a client to enter a new job in a queue.

The following fields within the NWQueueJobStruct\_t structure must be assigned values before this call can be made. (See Appendix A, <sup>a</sup>NWQueueJobStruct\_t Structure.)

targetServerID                    The objectID of the queue server or 0xFFFFFFFF for any server. The following example assigns 0xB0000012 as the targetServerID:

```
jobStruct.targetServerID = 0xB0000012;
```

targetExecutionTime            The time you want the file processed. The field is a 6 byte field with the following format: year, month, day, hour, minute, second. Use 0xFFFFFFFFFFFF for first opportunity. This example assigns May 1, 1991, 9:30:10 am as the targetExecutionTime:

```
jobStruct.targetExecutionTime[0] = 91;  
jobStruct.targetExecutionTime[1] = 5;  
jobStruct.targetExecutionTime[2] = 1;  
jobStruct.targetExecutionTime[3] = 9;  
jobStruct.targetExecutionTime[4] = 30;  
jobStruct.targetExecutionTime[5] = 10;
```

jobType                         The number representing the type of job serviced by the server; this number is server dependent. The following example assigns 0x00 as the jobType (0x00 means the queue server does not use this field):

```
jobStruct.jobType = 0;
```

jobControlFlags                The control flag that has been assigned to the job. Use any of the following:

```
NWCF_OPERATOR_HOLD
```

NWCF\_USER\_HOLD  
NWCF\_ENTRY\_OPEN  
NWCF\_SERVICE\_RESTART  
NWCF\_SERVICE\_AUTO\_START

The following example assigns NWCF\_SERVICE\_RESTART:

```
jobStruct.jobControlFlags =  
NWCF_SERVICE_RESTART;
```

**jobDescription**                      A string containing the content or purpose of the job.  
The following example assigns "Print Job" as the job description:

```
strcpy(jobStruct.jobDescription,  
"Print Job");
```

The queueRecord field may need to be filled in.

- If the file being submitted to the queue is a NetWare print job, the client must first allocate a printRecord and fill in the NWPrintRecord\_t Structure.
- Use NWConvertPrintStructToQueueStruct to fill in the queueRecord with the printRecord information. Then NWCreateQueueFile can be called.
- If the client wants to verify the printRecord after making this call, use NWConvertQueueStructToPrintStruct to convert the queueRecord field back into the printRecord.
- If the file being submitted to the queue is not a NetWare print job and the queue server uses the queueRecord parameter, the queue server must provide its own function to fill in the queueRecord parameter.
- If the queue server does not use the queueRecord parameter, the parameter does not need to be filled in.
- The file server fills in all other fields within the jobStruct parameter and returns it to the requesting client.
- The job will not be serviced until the file is closed with NWCloseFileAndStartQueueJob.

## Notes

This function can be used in conjunction with the NWReadQueueJobEntry function to change a portion of the job's entry record. However, if the target entry is already being serviced, NWChangeQueueJobEntry returns a Q\_SERVICING error and makes no changes to the job's entry record.

## See Also

NWChangeQueueJobEntry  
NWCloseFileAndAbortQueueJob  
NWCloseFileAndStartQueueJob  
NWConvertPrintStructToQueueStruct  
NWConvertQueueStructToPrintStruct  
NWRemoveJobFromQueue

## NWDestroyQueue



This function deletes a queue.

## Synopsis

```
#include "nwapi.h"

int                ccode;
uint16            serverConnID;
uint32            queueID;

ccode=NWDestroyQueue( serverConnID, queueID );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue to be deleted.

## Output

None.

## Return Values

- 0      Successful.
- 1     Unsuccessful. One of the following error codes is placed in NWErrno.
  - 0xFC    No Such Object
  - 0xF4    No Object Delete Privilege
  - 0x30    Invalid Connection ID
  - 0xFF    Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function destroys the queue specified by the queueID parameter. All active jobs are aborted, all servers are detached from the queue, and all jobs in the queue are destroyed and their associated files deleted. The queue object and its associated properties are removed from the bindery and the queue's subdirectory is deleted.

## Notes

Only SUPERVISOR or a bindery object that is security equivalent to SUPERVISOR can destroy a queue.

## See Also

NWCreateQueue

## NWDetachQueueServerFromQueue

This function removes the calling client from the queue's list of active queue servers.

## Synopsis

```
#include "nwapi.h"

int                ccode;
```

```

uint16      serverConnID;
uint32      queueID;

ccode=NWDetachQueueServerFromQueue( serverConnID, queueID );

```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue from which the calling station is being detached.

## Output

None.

## Return Values

0	Successful.	
-1	Unsuccessful.	One of the following error codes is placed in NWErrno:
	0x99 (153)	Directory Full
	0xD0 (208)	Queue Error
	0xD1 (209)	No Queue
	0xD2 (210)	No Queue Server
	0xD3 (211)	No Queue Rights
	0xD4 (212)	Queue Full
	0xD5 (213)	No Queue Job
	0xD6 (214)	No Job Right
	0xD7 (215)	Queue Servicing
	0xD8 (216)	Queue Not Active
	0xD9 (217)	Station Not Server
	0xDA (218)	Queue Halted
	0xDB (219)	Max. Queue Servers
	0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function removes the requesting client from the queue's list of active queue servers. If the requesting client is servicing a job, that service is automatically aborted.

## Notes

Only a workstation previously attached to the queue as a queue server can call this function.

## See Also

NWAttachQueueServerToQueue  
 NWReadQueueServerCurrentStatus  
 NWSetQueueServerCurrentStatus

## NWFinishServicingQueueJob

This function signals that a job has been completed successfully.

## Synopsis

```
#include "nwapi.h"
```

int	ccode;
uint16	serverConnID;
uint32	queueID;
uint16	jobNumber;
NWFileHandle_t	fileHandle;

```
ccode=NWFinishServicingQueueJob( serverConnID, queueID, jobNumber,
fileHandle );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue containing the job being finished.

*jobNumber* passes the job number of the job being finished.

*fileHandle* passes a pointer to the file handle for the file associated with the queue job.

## Output

None.

## Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

**Note:** Because this call uses NWCloseFile, it is possible to get NWErrno=0x001100xx. The 0x0011 indicates a file system error. See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows a queue server to signal the QMS that it has serviced a job successfully. The job entry is destroyed, and the job file is closed and deleted.

The calling queue server's access rights to the queue server are restored to their original (login) values.

## Notes

Only a queue server that has accepted a job to service can call this function.

**See Also**

NWAbortServicingQueueJob  
NWChangeToClientRights  
NWServiceQueueJob

**NWGetQueueJobFileSize**

This function returns the file size of the specified queue job.

**Synopsis**

```
#include "nwapi.h"

int          ccode;
uint16      serverConnID;
uint32      queueID;
uint16      jobNumber;
uint32      fileSize;

ccode=NWGetQueueJobFileSize( serverConnID, queueID, jobNumber,
&fileSize );
```

**Input**

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue to which the job is associated.

*jobNumber* passes the number of the job for which the information will be obtained.

*fileSize* passes a pointer to the space allocated for the file size.

**Output**

*fileSize* receives the file size.

**Return Values**

0	Successful.	
-1	Unsuccessful.	One of the following error codes is placed in NWErrno:
	0x99 (153)	Directory Full
	0xD0 (208)	Queue Error
	0xD1 (209)	No Queue
	0xD2 (210)	No Queue Server
	0xD3 (211)	No Queue Rights
	0xD4 (212)	Queue Full
	0xD5 (213)	No Queue Job
	0xD6 (214)	No Job Right
	0xD7 (215)	Queue Servicing
	0xD8 (216)	Queue Not Active
	0xD9 (217)	Station Not Server
	0xDA (218)	Queue Halted
	0xDB (219)	Max. Queue Servers
	0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

# NWGetQueueJobList

This function returns a list of all jobs associated with a given queue.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
uint16       numberOfJobsInQueue;
uint16       listOfJobNumbers[NWMAX_NUMBER_OF_
                                JOB_NUMBERS];

ccode=NWGetQueueJobList( serverConnID, queueID,
&numberOfJobsInQueue, listOfJobNumbers );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue whose job list is being reported.

*numberOfJobsInQueue* passes a pointer to the space allocated for the number of jobs in the queue.

*listOfJobNumbers* passes a pointer to the array allocated for the job numbers.

## Output

*numberOfJobsInQueue* receives the number of jobs currently in the Queue (0 - 250).

*listOfJobNumbers* receives the job numbers of all the jobs in the queue (0 - 250 numbers possible).

## Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:
0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows a program to get a list of all the jobs currently in a queue. When used in conjunction with the NWReadQueueJobEntry function, this function allows an application to retrieve information about all the jobs in a given queue. Because the QMS environment is multithreaded, however, the positioning, number and type of jobs in the queue can change between consecutive calls.

This function allows a workstation to determine how many jobs are in the queue at a particular instant and the job number of each. If a subsequent call to read information about a job in the queue fails with a NO\_Q\_JOB error, the requesting workstation can assume that either the job was deleted from the queue or its service was completed.

## Notes

The workstation making this call must be security equivalent to one of the objects listed in the queue's Q\_USERS or Q\_OPERATORS group properties.

## See Also

NWChangeQueueJobEntry  
NWChangeQueueJobPosition  
NWReadQueueJobEntry

## NWReadQueueCurrentStatus

This function returns the current status of a queue.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16      serverConnID;
uint32      queueID;
uint8       queueStatus;
uint16      numberOfJobsInQueue;
uint16      numberOfServers;
uint32      serverObjectIDList[NWMAX_NUMBER_
                                OF_SERVER_OBJECT_IDS];

uint16      clientConnIDList[NWMAX_NUMBER_
                                OF_SERVER_CONN_NUMBERS];

ccode=NWReadQueueCurrentStatus( serverConnID, queueID,
&queueStatus,&numberOfJobsInQueue, &numberOfServers,
serverObjectIDList, clientConnIDList );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue for which the status is being obtained.

*queueStatus* passes a pointer to the space allocated for the queue status.

*numberOfJobsInQueue* passes a pointer to the space allocated for the number of jobs in the queue.

*numberOfServers* passes a pointer to the space allocated for the number of attached queue servers.

*serverObjectIDList* passes a pointer to an array allocated for queue server object IDs associated with the numberOfServers parameter.

*clientConnIDList* passes a pointer to an array allocated for clientConnIDs corresponding to the servers returned by

the `serverObjectIDList` parameter.

## Output

*queueStatus* receives the status of the specified queue. (See Appendix A, "Queue Status Flags.")

*numberOfJobsInQueue* receives the number of jobs currently in the queue.

*numberOfServers* receives the number of attached queue servers.

*serverObjectIDList* receives an array of server IDs associated with the `numberOfServers` parameter.

*clientConnIDList* receives an array of `clientConnIDs` corresponding to the servers returned by the `serverObjectIDList` parameter.

## Return Values

0        Successful.  
-1       Unsuccessful. One of the following error codes is placed in `NWErrno`:

0xFC	No Such Object
0x30	Invalid Connection ID
0xFF	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in `NWErrno`.

## Description

This function is a queue server function which reads the current status of the specified queue. The `queueStatus` parameter indicates the overall status of the queue. (See Appendix A, "Queue Status Flags.")

The `numberOfJobsInQueue` parameter contains a count of the number of jobs currently in the queue, 0 to 250.

The `numberOfServers` parameter contains a count of the number of queue servers currently attached to service this queue, 0 to 25.

The `serverObjectIDList` and `clientConnIDList` parameters list queue servers currently servicing the queue by the queue server's `objectID` and the queue server's current workstation attachment (`clientConnID`).

## Notes

Workstations making this call must be security equivalent to one of the objects listed in the queue's `Q_USERS` or `Q_OPERATORS` group properties.

## See Also

`NWAttachQueueServerToQueue`  
`NWDetachQueueServerFromQueue`  
`NWReadQueueServerCurrentStatus`  
`NWSetQueueCurrentStatus`  
`NWSetQueueServerCurrentStatus`

## NWReadQueueJobEntry

This function retrieves information about a specified queue job.

## Synopsis

```
#include "nwapi.h"
```

```
int                ccode;  
uint16            serverConnID;  
uint32            queueID;  
uint16 j          obNumber;  
NWQueueJobStruct_t jobStruct;
```

```
ccode=NWReadQueueJobEntry( serverConnID, queueID, jobNumber,  
&jobStruct );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue associated with the queue job being read.

*jobNumber* passes the number of the job being read.

*jobStruct* passes a pointer to the structure (NWQueueJobStruct\_t) allocated for queue job information.

## Output

*jobStruct* receives the structure containing the queue job information. (See Appendix A, "NWQueueJobStruct\_t Structure.")

## Return Values

0        Successful.  
-1       Unsuccessful. One of the following error codes is placed in NWErrno:

0xD0	Queue Error
0xFC	No Such Object
0xD1	No Queue
0x30	Invalid Connection ID
0xD3	No Queue Rights
0xFF	Failure
0xD5	No Queue Job

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows an application to retrieve information about a job from a queue. The job's full 256-byte record is returned. ( See "NWQueueJobStruct\_t structure" in Appendix A.)

## Notes

Workstations making this call must be security equivalent to one of the objects listed in the queue's Q\_USER or Q\_OPERATORS group properties.

## See Also

NWChangeQueueJobEntry  
NWChangeQueueJobPosition  
NWCreateQueueFile  
NWGetQueueJobList

## NWReadQueueServerCurrentStatus



This function reads the current status of a queue server.

## Synopsis

```
#include "nwapi.h"
```

```
int                ccode;  
uint16            serverConnID;  
uint32            queueID;  
uint32            queueServerID;  
uint16            queueServerClientConnID;  
void              serverStatusRecord[NWMAX_SERVER_STATUS_  
                                RECORD_LENGTH];
```

```
ccode=NWReadQueueServerCurrentStatus( serverConnID, queueID,  queueServerID,  
queueServerClientConnID, serverStatusRecord );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue being affected.

*queueServerID* passes the bindery object ID of the queue server whose current status is being read.

*queueServerClientConnID* passes the connection number of the queue server being read.

*serverStatusRecord* passes a pointer to the buffer allocated for the status of the specified queue server.

## Output

*serverStatusRecord* receives the status of the specified queue server (64 bytes).

## Return Values

0        Successful.  
-1       Unsuccessful. One of the following error codes is placed in NWErrno:

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows a station to read the current status of a queue server. The QMS maintains a 64-byte status record for each queue server attached to a queue.

The QMS does not interpret the contents of the status record. The record contains information important to the calling application. We recommend that the first 4 bytes of this record contain an estimated price for the given server to complete a "standard" job.

Notes

Workstations making this call must be security equivalent to one of the objects listed in the queue's Q\_USER or Q\_OPERATORS group properties.

See Also

NWSetQueueServerCurrentStatus

NWRemoveJobFromQueue

This function removes a job from a queue.

Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
uint16       jobNumber;

ccode=NWRemoveJobFromQueue( serverConnID, queueID, jobNumber );
```

Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue where the job to be removed is located.

*jobNumber* passes the number of the job being removed.

Output

None.

Return Values

0        Successful.

-1       Unsuccessful. One of the following error codes is placed in NWErrno:

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows the workstation to remove a job from a queue. The jobNumber parameter contains the job number returned by the QMS when the job was created. The job number can also be obtained by using the NWGetQueueJobList function.

The specified job is removed from the queue, and the job file is closed and deleted. If the job is being serviced, the service is aborted. Further I/O requests made to the job's queue file return an ILLEGAL\_FILE\_HANDLE error.

## Notes

Both the job's creator and an operator can call this function.

## See Also

NWChangeQueueJobEntry  
 NWChangeQueueJobPosition  
 NWCreateQueueFile  
 NWGetQueueJobList  
 NWReadQueueJobEntry

## NWRestoreQueueServerRights

This function restores a server's own identity after it has assumed its client's rights.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;

ccode=NWRestoreQueueServerRights( serverConnID );
```

## Input

*serverConnID* passes the file server's connection ID.

## Output

None.

## Return Values

0 Successful.  
 -1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right

0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows a queue server to restore its own identity after it has assumed its client's identity using the NWChangeToClientRights function. The queue server's login user identification and associated security equivalence list are restored to its original values.

This function does not change any of the path mappings (directory bases) held by the queue server. However, access rights to those directories are adjusted to reflect the rights the queue server has in those directories.

If the queue server has changed some of its path mappings as part of its efforts to service the queue job, the queue server must restore those directory bases.

Files opened using the client's rights before this function is called continue to be accessible with the client's rights. Files opened after this function is called are accessible only with rights of the queue server.

## Notes

Only queue servers that have previously changed their identity using the NWChangeToClientRights function can call this function.

## See Also

NWChangeToClientRights

## NWServiceQueueJob

This function allows a queue server to select a new job for servicing.

## Synopsis

```
#include "nwapi.h"

int
uint16
uint32
uint16
NWQueueJobStruct_t
NWFileHandle_ta
ccode;
serverConnID;
queueID;
targetJobType;
jobStruct;
fileHandle;

ccode=NWServiceQueueJob( serverConnID, queueID, targetJobType,
&jobStruct, fileHandle );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue whose jobs are being serviced.

*targetJobType* passes the type of the job to be serviced.

*jobStruct* passes a pointer to the job record of the next available job returned by the QMS. (See Appendix A, <sup>a</sup>NWQueueJobStruct\_t Structure.)

*fileHandle* passes a pointer to the file handle for the file associated with the job to be serviced.

**Return Values**

0	Successful.	
-1	Unsuccessful.	One of the following error codes is placed in NWErrno:
	0x99 (153)	Directory Full
	0xD0 (208)	Queue Error
	0xD1 (209)	No Queue
	0xD2 (210)	No Queue Server
	0xD3 (211)	No Queue Rights
	0xD4 (212)	Queue Full
	0xD5 (213)	No Queue Job
	0xD6 (214)	No Job Right
	0xD7 (215)	Queue Servicing
	0xD8 (216)	Queue Not Active
	0xD9 (217)	Station Not Server
	0xDA (218)	Queue Halted
	0xDB (219)	Max. Queue Servers
	0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

**Description**

This function allows a queue server to select a new job for servicing.

**Notes**

The requesting client must have previously established itself as a queue server for the target queue.

**See Also**

- NWAbortServicingQueueJob
- NWAttachQueueServerToQueue
- NWCreateQueueFile
- NWFinishServicingQueueJob

**NWSetQueueCurrentStatus**

This function modifies a queue's status.

**Synopsis**

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
uint8        queueStatus;

ccode=NWSetQueueCurrentStatus( serverConnID, queueID, queueStatus );
```

**Input**

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue whose status is being updated.

*queueStatus* passes the control byte that determines the new status. (See Appendix A, "Queue Status Flags.")

## Output

None.

## Return Values

0 Successful.  
-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0xD3	No Queue Rights
0xFC	No Such Object
0x30	Invalid Connection ID
0xFF	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

## Description

This function allows the operator to control the addition of jobs and servers to the queue.

## Notes

The client making this call must be logged in as one of the objects listed in the Q\_OPERATORS property. The requesting client can become a queue operator by specifying its objectID when creating the queue (NWCreateQueue) or by adding its objectID with NWCreateProperty (see the Bindery Services chapter).

## See Also

NWAttachQueueServerToQueue  
NWCreateQueue  
NWDetachQueueServerFromQueue  
NWReadQueueCurrentStatus

## NWSetQueueServerCurrentStatus

This function updates QMS's copy of a server's status record.

## Synopsis

```
#include "nwapi.h"

int          ccode;
uint16       serverConnID;
uint32       queueID;
void         serverStatusRecord[NWMAX_SERVER_
                                STATUS_RECORD_LENGTH];

ccode=NWSetQueueServerCurrentStatus( serverConnID, queueID,
serverStatusRecord );
```

## Input

*serverConnID* passes the file server connection ID.

*queueID* passes the bindery object ID of the queue to which the specified queue server is attached.

*serverStatusRecord* passes a pointer to the buffer containing the new status record of the queue server (64 bytes).

**Output**

None.

**Return Values**

0           Successful.

-1          Unsuccessful.   One of the following error codes is placed in NWErrno:

0x99 (153)	Directory Full
0xD0 (208)	Queue Error
0xD1 (209)	No Queue
0xD2 (210)	No Queue Server
0xD3 (211)	No Queue Rights
0xD4 (212)	Queue Full
0xD5 (213)	No Queue Job
0xD6 (214)	No Job Right
0xD7 (215)	Queue Servicing
0xD8 (216)	Queue Not Active
0xD9 (217)	Station Not Server
0xDA (218)	Queue Halted
0xDB (219)	Max. Queue Servers
0xFF (255)	Failure

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

**Description**

The QMS does not interpret the contents of the status record. The record contains information important to the calling application only. We recommend that the first 4 bytes of this record contain an estimated price for the given server to complete a "standard" job.

**Notes**

Only workstations that have previously been attached to the queue as a queue server can make this call.

**See Also**

NWReadQueueServerCurrentStatus