# Synchronization Services APIs

## Introduction to Synchronization Services

Synchronization services calls enable applications to coordinate access to network files and other resources. These services are divided into two categories: Locking and Semaphores.

### Locking and Semaphores

NetWare provides calls that allow applications to lock files, physical records or logical records. The client must maintain tables which keep track of file and record locks. There are no APIs which can return this information.

Before locking a file/record, a client should also record the following information about the file/record in a File Log table residing on the file server:

- Name (for files)
- Location and size (for records)

There are two kinds of locks available to application developers-physical and logical. Unlike a physical record lock, a logical record lock does not actually lock bytes. Instead, a logical record lock acts somewhat like a semaphore. Applications cooperatively define a logical record name that represents a group of files, records, structures and so on. When an application locks a logical record, it only locks the logical record name, not the group of files, records or structures the name represents. Any uncooperative application can ignore a lock on the logical record name and directly lock the physical files or records. Therefore, applications using logical record locks should not use other locking techniques simultaneously on the same data.

NetWare also provides calls that enable applications to create, open, examine and close semaphores. Applications can also use semaphore calls to increment and decrement the value associated with a semaphore.

## NWClearFile

This function unlocks the specified file and removes it from the File Log table.

## Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;

ccode=NWClearFile( &path );
```

## Input

*path* passes a pointer to the name of the file whose file lock is being cleared.

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |

| | |
|---|---|
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

If the file is open, this function closes it and invalidates the file handle. The file can then be accessed by other clients. If the calling client had the file open multiple times, it is closed as many times and all associated file handles are invalidated.

### Notes

The fileName parameter can specify either a file's complete path name or a path relative to the default directory mapping.

### See Also

NWClearFileSet

## NWClearFileSet

This function unlocks all the files logged for a specified client.

### Synopsis

```
#include "nwapi.h"
int                     ccode;

ccode=NWClearFileSet( );
```

### Input

None.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xF | FLock Error |
| 0xFF | Unlock Error |

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

### Description

All open files in the client's File Log table are closed and the file handles are cleared. This includes files logged across any servers that have been logged into the set.

### Description

This function is ignored if the requesting client does not have logged files.

## See Also

NWClearFile
NWLockFileSet
NWLogFile
NWReleaseFile
NWReleaseFileSet

## NWClearLogicalRecord

This function unlocks a logical record and removes it from the logical record log table.

## Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;
char                    logicalRecordName[NWMAX_LOGICAL_
                                          RECORD_NAME_LENGTH];

ccode=NWClearLogicalRecord( serverConnID, logicalRecordName );
```

## Input

*serverConnID* passes the file server connection ID.

*logicalRecordName* passes a pointer to the name of the logical record being cleared (128 characters).

## Output

None.

## Return Values

| | | |
|---|---|---|
| 0 | Successful. | |
| -1 | Unsuccessful. | One of the following error codes is placed in NWErrno: |
| | 0xFD | Lock Collision |
| | 0xFE | Timeout Error |
| | 0xFF | Lock Error |
| | 0xFF | Unlock Error |

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

## Description

This function clears a record residing in the Logical Record Log table associated exclusively with the requesting client.

Applications define logical record names. A logical record name represents a group of files, physical records, structures and so on. The NWLogLogicalRecord or NWLockLogicalRecordSet functions locks one or more logical record names, not the actual files, physical records or structures associated with each logical record name. Any uncooperative application can ignore a lock on the logical record name and directly lock physical files or records. Therefore, applications using logical record locks must not use other locking techniques simultaneously.

## See Also

NWClearLogicalRecordSet

NWLockLogicalRecordSet
NWLogLogicalRecord
NWReleaseLogicalRecord
NWReleaseLogicalRecordSet

## NWClearLogicalRecordSet

This function unlocks and removes all logical records in the Logical Record Log table (on any of the servers you are attached to).

### Synopsis

```
#include "nwapi.h"

int                     ccode;

ccode=NWClearLogicalRecordSet( );
```

### Input

None.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

This function unlocks all of the logical records that are logged in the Logical Record Log table and removes them from the table. If the requesting process does not have logged logical records, this function is ignored.

### See Also

NWClearLogicalRecord
NWLockLogicalRecordSet
NWLogLogicalRecord
NWReleaseLogicalRecord
NWReleaseLogicalRecordSet

## NWClearPhysicalRecord

This function unlocks the specified physical record and removes it from the Log table.

### Synopsis

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

```
int             ccode;
uint16          serverConnID;
NWFileHandle_ta fileHandle;
uint32          recordStartOffset;
uint32          recordLength;

ccode=NWClearPhysicalRecord( serverConnID, fileHandle, recordStartOffset, recordLength);
```

### Input

*serverConnID* passes the file server connection ID.

*fileHandle* passes the file handle associated with the file containing the physical record being cleared.

*recordStartOffset* passes the offset, within the file, where the locked record begins.

*recordLength* passes the length, in bytes, of the locked record.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

The Record Log table resides on the file server and is associated exclusively with the requesting task on the workstation.

The application locates the physical record within the specified file by passing the offset in the recordStartOffset parameter and the length in the recordLength parameter.

### Notes

This function is ignored if the requesting workstation does not have logged physical records.

### See Also

NWClearPhysicalRecordSet
NWLockPhysicalRecordSet
NWLogPhysicalRecord
NWReleasePhysicalRecord
NWReleasePhysicalRecordSet

## NWClearPhysicalRecordSet

This function unlocks and removes all physical records from the Physical Record Log table.

### Synopsis

```
#include "nwapi.h"

int                     ccode;

ccode=NWClearPhysicalRecordSet( );
```

### Input

None.

### Output

None.

### Return Values

| | | |
|---|---|---|
| 0 | Successful. | |
| -1 | Unsuccessful. | One of the following error codes is placed in NWErrno: |
| | 0xFD | Lock Collision |
| | 0xFE | Timeout Error |
| | 0xFF | Lock Error |
| | 0xFF | Unlock Error |

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

### Description

This function clears an entire set of physical records associated with the requesting client.

### Notes

This function is ignored if the requesting workstation does not have logged physical records.

### See Also

NWClearPhysicalRecord
NWLockPhysicalRecordSet
NWLogPhysicalRecord
NWReleasePhysicalRecord
NWReleasePhysicalRecordSet

## NWCloseSemaphore

This function closes a semaphore.

### Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;
uint32                  NetWareSemaphoreHandle;

ccode=NWCloseSemaphore( serverConnID, NetWareSemaphoreHandle );
```

### Input

*serverConnID* passes the file server connection ID.

*NetWareSemaphoreHandle* passes the semaphore handle obtained when the semaphore was opened (NWOpenSemaphore).

### Output

None.

### Return Values

| | | |
|---|---|---|
| 0 | Successful. | |
| -1 | Unsuccessful. | One of the following error codes is placed in NWErrno: |
| | 0xFD | Lock Collision |
| | 0xFE | Timeout Error |
| | 0xFF | Lock Error |
| | 0xFF | Unlock Error |

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

### Description

This function decrements the open count of the semaphore, indicating that one less process is holding the semaphore open. If the requesting process is the last process to have this semaphore open, the semaphore is deleted.

### See Also

NWExamineSemaphore
NWOpenSemaphore
NWSignalSemaphore
NWWaitOnSemaphore

## NWExamineSemaphore

This function returns the current value and open count for the specified semaphore.

### Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
uint32          semaphoreHandle;
int             semaphoreValue;
uint16          semaphoreOpenCount;

ccode=NWExamineSemaphore( serverConnID, semaphoreHandle,
&semaphoreValue, &semaphoreOpenCount );
```

### Input

*serverConnID* passes the file server connection ID.

*semaphoreHandle* passes the semaphore handle obtained when the semaphore was opened (NWOpenSemaphore).

*semaphoreValue* passes a pointer to the space allocated for the current semaphore value (optional).

*semaphoreOpenCount* passes a pointer to the space allocated for the number of stations that currently have this semaphore open.

### Output

*semaphoreValue* receives the current semaphore value (optional).

*semaphoreOpenCount* receives the number of stations that currently have this semaphore open.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

The semaphore value is decremented for each NWWaitOnSemaphore function call and incremented for each NWSignalSemaphore function call.

A positive semaphore value indicates that the application can access the associated network resource. A negative value indicates the number of processes waiting to use the semaphore. If the semaphore value is negative, the application must either enter a waiting queue by calling the function NWWaitOnSemaphore or temporarily abandon its attempt to access the network resource.

The semaphoreOpenCount parameter indicates the number of processes holding the semaphore open. A call to NWOpenSemaphore increments this value. A call to NWCloseSemaphore decrements this value.

### See Also

NWCloseSemaphore
NWOpenSemaphore
NWSignalSemaphore
NWWaitOnSemaphore

## NWLockFileSet

This function locks all files that are logged in the File Log table.

### Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  timeOutLimit;

ccode=NWLockFileSet( timeOutLimit );
```

### Input

*timeOutLimit* passes the length of time the file server waits.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

The timeOutLimit indicates how long the file server will attempt to lock the set. This limit is specified in units of 1/18 of a second (0 = no wait).

### See Also

NWClearFile
NWClearFileSet
NWLogFile
NWReleaseFile
NWReleaseFileSet

## NWLockLogicalRecordSet

This function locks all logical records logged in the Logical Record Log table.

### Synopsis

```
#include "nwapi.h"

int             ccode;
uint8           lockFlags;
uint16          timeOutLimit;

ccode=NWLockLogicalRecordSet( lockFlags, timeOutLimit );
```

### Input

*lockFlags* passes one of the following lock flags:

NWLS_EXCLUSIVE
NWLS_SHAREABLE

*timeOutLimit* passes the length of time the file server attempts to lock the record set before timing out.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |

| | |
|---|---|
| 0xFF | Unlock Error |

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

## Description

Applications define logical record names. A logical record name represents a group of files, physical records or data structures. NWLogLogicalRecord or NWLockLogicalRecordSet function affects one or more logical record names, not the actual files, physical records or data structures associated with each logical record name. Any uncooperative application can ignore a lock on the logical record name and directly lock physical files or records. Therefore, applications using logical record locks must not simultaneously use other locking techniques.

The timeOutLimit parameter indicates how long the file server will attempt to lock the logical record set. The timeOutLimit parameter is specified in units of
1/18 of a second (0 = no wait).

## See Also

NWClearLogicalRecord
NWClearLogicalRecordSet
NWLogLogicalRecord
NWReleaseLogicalRecord
NWReleaseLogicalRecordSet

## NWLockPhysicalRecordSet

This function locks all records logged in the Physical Record Log table.

## Synopsis

```
#include "nwapi.h"

int             ccode;
uint8           lockFlags;
uint16          timeOutLimit;

ccode=NWLockPhysicalRecordSet( lockFlags, timeOutLimit );
```

## Input

*lockFlags* passes one of the following lock flags:

NWLS_EXCLUSIVE
NWLS_SHAREABLE

*timeOutLimit* passes the length of time the file server attempts to lock the physical record set before timing out.

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |

| | |
|---|---|
| 0xFF | Unlock Error |

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

### Description

The timeOutLimit is specified in units of 1/18 of a second (0 = no wait).

### Notes

This function cannot lock a record that is already locked exclusively by another application. If one or more records, identified in the log table, are already exclusively locked by another application, the attempt to lock the set fails.

### See Also

NWClearPhysicalRecord
NWClearPhysicalRecordSet
NWLogPhysicalRecord
NWReleasePhysicalRecord
NWReleasePhysicalRecordSet

## NWLogFile

This function logs the file in the File Log table. Use this function when you need to lock a set of files. If the lock flag is set, this function also locks the file.

### Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint8           logFlags;
uint16          timeOutLimit;

ccode=NWLogFile( &path, logFlags, timeOutLimit );
```

### Input

*path* passes a pointer to the file name to be locked (must be the full path name for the file-up to 255 characters).

*logFlags* passes one of the following log flags.

NWFL_LOG_ONLY
NWFL_LOG_AND_LOCK

*timeOutLimit* passes the length of time the file server attempts to log the specified file before timing out.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |

| | |
|---|---|
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

## Description

When the logFlags parameter is set to NWFL_LOG_AND_LOCK, the server will attempt to lock a file for the length of time specified by the timeOutLimit parameter. The timeOutLimit parameter is specified in units of 1/18 of a second (0 = no wait).

A log table contains data locking information used by a file server. The file server tracks this information for each workstation and process. Whenever a file, logical record or physical record is logged, information identifying the data being logged is placed in the log table. Normally a set of files or records is logged and then locked as a set. However, a single file or record can also be locked when it is placed in the log table.

When using log tables a task first logs all of the files or records that are needed to complete a transaction. The task then attempts to lock the logged set of files or records. If some of the logged resources cannot be locked, the lock fails and none of the resources are locked.

## See Also

NWClearFile
NWClearFileSet
NWLockFileSet
NWReleaseFile

## NWLogLogicalRecord

This function logs a logical record.

## Synopsis

```
#include "nwapi.h"

int       ccode;
uint16    serverConnID;
char      logicalRecordName[NWMAX_LOGICAL_
                            RECORD_NAME_LENGTH];
uint8     lockFlags;
uint16    timeOutLimit;

ccode=NWLogLogicalRecord( serverConnID, logicalRecordName, lockFlags, timeOutLimit );
```

## Input

*serverConnID* passes the file server connection ID.

*logicalRecordName* passes a pointer to the name of the logical record being logged (128 characters).

*lockFlags* passes one of the following lock flags.

NWPL_LOG_ONLY
NWPL_LOG_AND_LOCK_EXCLUSIVE
NWPL_LOG_AND_LOCK_SHAREABLE

*timeOutLimit* passes the length of time the file server attempts to lock the record before timing out.

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

## Description

When the lockFlags parameter is set to option one or three, the file server will attempt to lock the logical record for the length of time specified by the timeOutLimit parameter. The timeOutLimit parameter is specified in units of 1/18 of a second.

A log table contains data locking information used by a file server. The file server tracks this information for each workstation and workstation task. Whenever a file, logical record or physical record is logged, information identifying the data being logged is placed in the log table.

Normally a set of files or records is logged and then locked as a set. However, a single file or record can also be locked when it is placed in the log table. The release functions are used to unlock a lock (or set of locks). The clear functions are used to unlock and remove a lock (or set of locks) from the log table.

When using log tables, a task first logs all files or records to complete a transaction. The task attempts to lock the logged set of files or records. If some logged resources cannot be locked, the lock fails and none of the resources are locked.

## See Also

NWClearLogicalRecord
NWClearLogicalRecordSet
NWLockLogicalRecordSet
NWReleaseLogicalRecord
NWReleaseLogicalRecordSet

## NWLogPhysicalRecord

This function logs a physical record in the Physical Record Log table.

## Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
NWFileHandle_ta fileHandle;
uint32          recordStartOffset;
uint32          recordLength;
uint8           lockFlags;
uint16          timeOutLimit;

ccode=NWLogPhysicalRecord( serverConnID, fileHandle, recordStartOffset, recordLength, lockFlags,
```

timeOutLimit );

## Input

*serverConnID* passes the file server connection ID.

*fileHandle* passes the file handle of the file whose record is being logged.

*recordStartOffset* passes the offset into the file where the record being logged begins.

*recordLength* passes the length, in bytes, of the record to be logged.

*lockFlags* passes one of the following lock flags:

NWPL_LOG_ONLY
NWPL_LOG_AND_LOCK_EXCLUSIVE
NWPL_LOG_AND_LOCK_SHAREABLE

*timeOutLimit* passes the length of time the file server attempts to log a record before timing out.

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

## Description

The file server attempts to log the record for the length of time specified by the timeOutLimit parameter before returning a time out error. The timeoutLimit parameter is specified in units of 1/18 of a second (0 = no wait).

A log table contains data locking information used by a file server. The file server tracks this information for each workstation and process. Whenever a file, logical record or physical record is logged, information identifying the data being logged is placed in the log table. Normally a set of files or records is logged and then locked as a set. However, a single file or record can also be locked when it is placed in the log table.

The release functions are used to unlock a lock or set of locks.

The clear functions are used to unlock and remove a lock or set of locks from the log table.

## Notes

When using log tables, a task first logs all files or records to complete a transaction. The task then attempts to lock the logged set of files or records. If some of the logged resources cannot be locked, the lock fails and none of the resources are locked.

## See Also

NWClearLogicalRecord
NWClearLogicalRecordSet

NWLockLogicalRecordSet
NWReleaseLogicalRecord
NWReleaseLogicalRecordSet

## NWOpenSemaphore

This function opens a semaphore.

## Synopsis

#include "nwapi.h"

```
int             ccode;
uint16          serverConnID;
char            semaphoreName[NWMAX_SEMAPHORE_NAME];
int             initialSemaphoreValue;
uint32          NetWareSemaphoreHandle;
uint16          semaphoreOpenCount;

ccode=NWOpenSemaphore( serverConnID, semaphoreName,
initialSemaphoreValue, &NetWareSemaphoreHandle, &semaphoreOpenCount );
```

## Input

*serverConnID* passes the file server connection ID.

*semaphoreName* passes a pointer to the name of the semaphore to be opened.

*initialSemaphoreValue* passes the initial value of the semaphore being opened (must be greater than 1).

*NetWareSemaphoreHandle* passes a pointer to the space allocated for the NetWare semaphore handle.

*semaphoreOpenCount* passes a pointer to the space allocated for the number of stations that have this semaphore opened (optional).

## Output

*NetWareSemaphoreHandle* receives the NetWare semaphore handle.

*semaphoreOpenCount* receives the number of stations that have this semaphore opened (optional).

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

## Description

This function creates and initializes the semaphore to the value indicated by the initialSemaphoreValue parameter.

The semaphoreOpenCount parameter indicates the number of processes currently using the semaphore.

Semaphores can have two purposes under NetWare.

1 Semaphores can limit the number of users to a particular resource. To limit the number of users, use the NWOpenSemaphore and NWCloseSemaphore function calls only.

The NWOpenSemaphore call returns the number of processes that currently have the semaphore open. An application can check the value against a programmer-defined limit and take appropriate action.

2 Semaphores can restrict access to a particular resource. If access is restricted, only serial access is allowed. To request access, a resource must open the semaphore associated with the resource and, through that semaphore, request permission to access the resource. If the resource is unavailable, the calling process is placed in a wait queue.

Restricting access to a resource is not limited to allowing only one process to have access. For example, if the resource being restricted is a modem pool with four modems, the semaphore could allow access to four users but could restrict access to subsequent requesters by setting the initialSemaphoreValue parameter to four.

## See Also

NWCloseSemaphore
NWExamineSemaphore
NWSignalSemaphore
NWWaitOnSemaphore

## NWReleaseFile

This function unlocks the specified file but does not remove it from the File Log table.

## Synopsis

```
#include "nwapi.h"

int                     ccode;
NWPath_t                path;

ccode=NWReleaseFile( &path );
```

## Input

*path* passes a pointer to the NWPath_t structure containing the directory handle, the server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |

| | |
|---|---|
| 0xFF | Unlock Error |

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

### See Also

NWClearFile
NWClearFileSet
NWLogFile
NWReleaseFileSet

NWReleaseFileSet

## NWReleaseFileSet

This function unlocks all files logged in the File Log table.

### Synopsis

```
#include "nwapi.h"
int                     ccode;

ccode=NWReleaseFileSet( );
```

### Input

None.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

The NWReleaseFileSet function does not remove files from the log table. This function is ignored if the requesting workstation does not have locked files.

### See Also

NWClearFile
NWClearFileSet
NWLogFile
NWReleaseFile

## NWReleaseLogicalRecord

This function unlocks a logical record but does not remove it from the Logical Record Log table.

## Synopsis

#include "nwapi.h"

```
int             ccode;
uint16          serverConnID;
char            logicalRecordName[NWMAX_LOGICAL_
                RECORD_NAME];
```

ccode=**NWReleaseLogicalRecord**( serverConnID, logicalRecordName );

## Input

*serverConnID* passes the file server connection ID.

*logicalRecordName* passes a pointer to the name of the logical record being released (128 characters).

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

## Description

A log table contains data locking information used by a file server. The file server tracks this information for each workstation and workstation task. Whenever a file, logical record or physical record is logged, information identifying the data being logged is placed in the log table. Normally a set of files or records is logged and then locked as a set. However, a single file or record can also be locked when it is placed in the log table.

## See Also

NWClearLogicalRecord

# NWReleaseLogicalRecordSet

This function unlocks all the logical records in the Logical Record Log table.

## Synopsis

#include "nwapi.h"

```
int             ccode;
```

ccode=**NWReleaseLogicalRecordSet**( );

## Input

None.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

The NWReleaseLogicalRecordSet function does not remove logical records from the log table. This function is ignored if the requesting workstation or process does not have locked logical records.

### See Also

NWClearLogicalRecord
NWClearLogicalRecordSet
NWLockLogicalRecordSet
NWLogLogicalRecord
NWReleaseLogicalRecord

## NWReleasePhysicalRecord

This function unlocks the specified physical record but does not remove it from the log table.

### Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
NWFileHandle_ta fileHandle;
uint32          recordStartOffset;
uint32          recordLength;

ccode=NWReleasePhysicalRecord( serverConnID, fileHandle,
recordStartOffset, recordLength );
```

### Input

*serverConnID* passes the file server connection ID.

*fileHandle* passes the file handle associated with the file containing the specified record.

*recordStartOffset* passes the offset, within the file, where the physical record begins.

*recordLength* passes the length, in bytes, of the record being released.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### See Also

NWClearPhysicalRecord
NWClearPhysicalRecordSet
NWLockPhysicalRecordSet
NWLogPhysicalRecord
NWReleasePhysicalRecordSet

## NWReleasePhysicalRecordSet

This function unlocks all records logged in the Physical Record Log table but leaves them logged in the table.

### Synopsis

```
#include "nwapi.h"

int                     ccode;

ccode=NWReleasePhysicalRecordSet( );
```

### Input

None.

### Output

None.

### Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

### Description

This function is ignored if the workstation does not have locked physical records.

### See Also

NWClearPhysicalRecord

NWClearPhysicalRecordSet
NWLockPhysicalRecordSet
NWLogPhysicalRecord
NWReleasePhysicalRecord

## NWSignalSemaphore

This function signals a semaphore that the station or process is finished.

## Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
uint32          semaphoreHandle;

ccode=NWSignalSemaphore( serverConnID, semaphoreHandle );
```

## Input

*serverConnID* passes the file server connection ID.

*NetWareSemaphoreHandle* passes the semaphore handle pointing to the semaphore to be signaled.

## Output

None.

## Return Values

| | | |
|---|---|---|
| 0 | Successful. | |
| -1 | Unsuccessful. | One of the following error codes is placed in NWErrno: |
| | 0xFD | Lock Collision |
| | 0xFE | Timeout Error |
| | 0xFF | Lock Error |
| | 0xFF | Unlock Error |

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

## Description

An application must call this function when it finishes accessing the network resource associated with the semaphore. If processes are waiting to use the semaphore, the first process in the queue is released (signaled). The NetWareSemaphoreHandle is a uint32 pointer to a semaphore. An application obtains this handle with a call to the NWOpenSemaphore function.

## See Also

NWCloseSemaphore
NWExamineSemaphore
NWOpenSemaphore
NWWaitOnSemaphore

## NWWaitOnSemaphore

This function decrements a semaphore value.

## Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;
uint32                  semaphoreHandle;
uint16                  timeOutLimit;

ccode=NWWaitOnSemaphore( serverConnID, semaphoreHandle,
timeOutLimit );
```

## Input

*serverConnID* passes the file server connection ID

*NetWareSemaphoreHandle* passes the semaphore handle returned from the NWOpenSemaphore function call.

*timeOutLimit* passes the length of time the application will wait.

## Output

None.

## Return Values

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

| | |
|---|---|
| 0xFD | Lock Collision |
| 0xFE | Timeout Error |
| 0xFF | Lock Error |
| 0xFF | Unlock Error |

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

## Description

If the semaphore value is greater than or equal to zero, the application can access the associated resource. If the value is less than zero, the function queues the application for the time interval specified in the timeoutLimit parameter. If the semaphore value is greater than or equal to zero when the timeout value expires, the application can access the resource. If the semaphore value is still negative, the function removes the application from the queue and re-increments the semaphore value. A semaphore handle is a uint32 pointer to a semaphore and can be obtained by calling NWOpenSemaphore.

The timeout limit indicates how long the file server should wait if the semaphore value is negative. The timeout limit is specified in units of 1/18 of a second (0 = no wait).

## See Also

NWCloseSemaphore
NWExamineSemaphore
NWOpenSemaphore
NWSignalSemaphore