Transaction Tracking System APIs

Introduction to Transaction Tracking

NetWare file servers that include the Transaction Tracking System (TTS) can track transactions and ensure file integrity by backing out (or erasing) interrupted or partially completed transactions. TTS only affects transactional files. A file becomes transactional when the file's Transactional extended file attribute is set.

For example, a banking database application frequently performs a transaction that includes the following three writes to database files:

- A debit to one account
- A credit to another account
- A note to a log

The application must complete all three of these writes to maintain database integrity. Transaction tracking is implemented in two ways, implicit and explicit.

- **Implicit transaction tracking** requires no coding on the part of an application developer. If TTS is installed and enabled on a file server, TTS tracks all transactions to all transactional files (including transactions made by NetWare to bindery files).

- **Explicit transaction tracking** has two calls: NWTTSBeginTransaction and NWTTSEndTransaction. Explicit Transaction Tracking requires applications to make TTS calls and allows applications to neatly bracket file update sequences with locking and TTS calls. An application would most likely use logical or physical record locks with TTS calls (see "Synchronization Services").

The following steps describe how TTS tracks each write within a transaction.

1) An application writes new data to a file on a file server.

2) The file server stores the new data in cache memory. The target file on the file server hard disk remains unchanged.

3) The file server scans the target file on the file server hard disk, finds the data to be changed (old data), and copies the old data to cache memory. The file server also records the name and directory path of the target file and the location and length of the old data (record) within the file. The target file on the file server hard disk still remains unchanged.

4) The file server writes the old data in cache memory to a transaction work file on the file server hard disk. The transaction work file resides at the root level of volume SYS on the file server. The file is flagged System and Hidden. The target file on the file server hard disk still remains unchanged.

5) The file server writes the new data in cache memory to the target file on the file server hard disk. The target file is now changed.

The file server repeats these steps for each write within a transaction. The transaction work file grows to accommodate the old data for each write. If the transaction is interrupted, the file server writes the contents of the transaction work file to the target file, thereby restoring the file to its pretransaction condition. In effect, the file server backs out the transaction.

A file server can monitor from 100 to 10,000 transactions at a time. (The maximum value can be configured with SET for NetWare v3.x.) A file server can track only one transaction at a time for each session. If a session sends several transactions to a file server rapidly, the file server queues the transactions and services them one at a time.

NWTTSAbortTransaction

This function aborts all transactions, explicit and implicit, on a file that has been flagged transactional.

Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;

ccode=NWTTSAbortTransaction( serverConnID );
```

Input

serverConnID passes the file server connection ID.

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

Note: A return value in NWErrno of 0xFE indicates that more than the threshold number of logical or physical records are still locked by the application. However, the transaction is still finished and any locks being held are released. In this case, the file server automatically starts a new implicit transaction.

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

Description

This function releases the following record locks:

- Physical record locks generated by the file server when an application tried to write an unlocked record.
- Physical or logical locks that have not been released because of a file write.

When this function is complete, all transactions will have been successfully backed out.

Notes

If a transaction is aborted, all writes made since the beginning of a transaction are cancelled, and all files are returned to the state they were in before the transaction began.

Files can be flagged transactional with NWCreateFile and NWSetFileAttributes.

See Also

NWCreateFile
NWSetFileAttributes
NWTTSBeginTransaction
NWTTSEndTransaction

NWTTSBeginTransaction

This function begins an explicit transaction on a file that has been flagged transactional.

Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;

ccode=NWTTSBeginTransaction( serverConnID );
```

Input

serverConnID passes the file server connection ID.

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

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

Description

This function tracks all transactional files that are currently open and all those that are opened during the transaction.

When data is written to a transactional file during a transaction, the file server automatically generates a physical record lock for the region being written. If a lock already exists, no additional lock is generated. This automatic locking can be disabled using the NWTTSSetControlFlags function.

Notes

Any closing and unlocking of transactional files is delayed until an NWTTSEndTransaction or NWTTSAbortTransaction is executed. Logical and physical records are not unlocked until the end of the transaction if file writes are performed while the lock is in force. Use NWCreateFile or NWSetFileAttributes to flag a file transactional.

See Also

NWCreateFile
NWSetFileAttributes
NWTTSAbortTransaction
NWTTSSetControlFlags
NWTTSEndTransaction

NWTTSDisableTransactionTracking

This function disables transaction tracking services on the specified file server.

Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;

ccode=NWTTSDisableTransactionTracking( serverConnID );
```

Input

serverConnID passes the file server connection ID.

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active
0xC6	No Console Privileges

Note: A return value in NWErrno of 0xFE indicates that more than the threshold number of logical or physical records are still locked by the application. However, the transaction is still finished and any locks being held are released. In this case, the file server automatically starts a new implicit transaction.

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

Description

This function call should be used after transaction services are no longer being used.

The client making this call must be supervisor or have equivalent rights.

See Also

NWTTSEnableTransactionTracking
NWTTSBeginTransaction
NWTTSIsTransactionWritten

NWTTSEnableTransactionTracking

This function enables transaction tracking services on the specified file server.

Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;

ccode=NWTTSEnableTransactionTracking( serverConnID );
```

Input

serverConnID passes the file server connection ID.

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active
0xC6	No Console Privileges

Note: A return value in NWErrno of 0xFE indicates that more than the threshold number of logical or physical records are still locked by the application. However, the transaction is still finished and any locks being held are released. In this case, the file server automatically starts a new implicit transaction.

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

Description

This function call will enable TTS if the server has it available. However, the version of NetWare that the file server is running determines the information the call returns.

For NetWare v3.*x*, this call enables TTS. Use NWTTSIsAvailable to check whether TTS has been disabled.

For NetWare v2.*x*, this call enables TTS if TTS has been installed on the file server. If TTS has not been installed, the call will not fail. On file servers running NetWare v2.*x*, you should always use NWTTSIsAvailable before making this call.

Notes

The application making this call must be supervisor or have equivalent rights.

See Also

NWTTSDisableTransactionTracking
NWTTSBeginTransaction
NWTTSIsAvailable
NWTTSIsTransactionWritten

NWTTSEndTransaction

This function ends an explicit transaction on a file that has been flagged transactional. The function also returns a transaction reference number.

Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;
uint32                  transactionReferenceNumber;
```

ccode=**NWTTSEndTransaction**(serverConnID,
&transactionReferenceNumber);

Input

serverConnID passes the file server connection ID.

transactionReferenceNumber passes a pointer to the space allocated for the transaction reference number for the transaction being ended.

Output

transactionReferenceNumber receives the transaction reference number for the transaction being ended.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

Note: A return value in NWErrno of 0xFE indicates that more than the threshold number of logical or physical records are still locked by the application. However, the transaction is still finished and any locks being held are released. In this case, the file server automatically starts a new implicit transaction.

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

Description

The transaction is not necessarily written to disk when the reference number is returned; a client must use the NWTTSIsTransactionWritten function to verify that a transaction has been written to disk. If the file server fails before all updates contained within the transaction have been written to disk, the transaction will be backed out when the file server is rebooted.

If transaction tracking is disabled, the reference number can still be used to determine when the transaction has been completely written to disk.

Notes

This function releases all physical record locks generated by the file server when a write is made to an unlocked record. In addition, physical or logical locks that have not been released because of a file write are unlocked at this time.

Files can be flagged transactional with NWCreateFile and with NWSetFileAttributes.

See Also

NWCreateFile
NWSetFileAttributes
NWTTSAbortTransaction
NWTTSBeginTransaction
NWTTSIsTransactionWritten

NWTTSGetConnectionThresholds

This function returns the number of logical and physical record locks allowed for implicit transactions.

Synopsis

```
#include "nwapi.h"

int                     ccode;
uint16                  serverConnID;
uint8                   logicalRecordLockThreshold;
uint8                   physicalRecordLockThreshold;

ccode=NWTTSGetConnectionThresholds( serverConnID,
&logicalRecordLockThreshold, &physicalRecordLockThreshold );
```

Input

serverConnID passes the file server connection ID.

logicalRecordLockThreshold passes a pointer to the space allocated for the number of logical record locks allowed before implicit transactions begin (0 to 255).

physicalRecordLockThreshold passes a pointer to the space allocated for the number of physical record locks allowed before implicit transactions begin (0 to 255).

Output

logicalRecordLockThreshold receives the number of logical record locks allowed before implicit transactions begin (0 to 255).

physicalRecordLockThreshold receives the number of physical record locks allowed before implicit transactions begin (0 to 255).

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

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 get the number of logical and physical record locks allowed before implicit transactions begin.

The NWTTSSetConnectionThresholds function and this function are useful for applications that change the implicit application thresholds and later want to restore them. For example, NWTTSGetConnectionThresholds can get the number of logical and physical locks, and NWTTSSetConnectionThresholds can do one of the following:

- Turn off implicit transactions. (Applications that use only explicit transactions, but sometimes generate unnecessary implicit transactions, need to turn off all implicit transactions.)
- Set implicit thresholds for applications that always keep one or more records locked.

Notes

The default threshold for logical and physical locks is 0. A threshold of 255 means implicit transactions for that lock type have been completed.

See Also

NWTTSSetConnectionThresholds

NWTTSGetControlFlags

This function returns the control flags byte for files flagged as transactional.

Synopsis

#include "nwapi.h"

```
int             ccode;
uint16          serverConnID;
uint8           TTSControlFlags;

ccode=NWTTSGetControlFlags( serverConnID, &TTSControlFlags );
```

Input

serverConnID passes the file server connection ID.

TTSControlFlags passes a pointer to the space allocated for the Transaction Tracking Control flags. (See ªDescriptionº below.)

Output

TTSControlFlags receives a Transaction Tracking Control flag. (See "Description" below.)

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

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

Description

Transaction tracking control flags are only valid for files flagged as TTS (transactional). These control flags are defined as follows:

0x00	Automatic record locking is disabled
0x01	Automatic record locking is enabled

See Also

NWTTSSetControlFlags

NWTTSGetProcessThresholds

This function returns the number of explicit physical and logical record locks that can be done before implicit locking begins.

Synopsis

```
#include "nwapi.h"

int         ccode;
uint16      serverConnID;
uint8       logicalRecordLockThreshold;
uint8       physicalRecordLockThreshold;

ccode=NWTTSGetProcessThresholds( serverConnID,
&logicalRecordLockThreshold, &physicalRecordLockThreshold );
```

Input

serverConnID passes the file server connection ID.

logicalRecordLockThreshold passes a pointer to space allocated for the number of explicit logical record locks allowed before implicit transactions begin.

physicalRecordLockThreshold passes a pointer to the space allocated for the number of explicit physical record locks allowed before implicit transactions begin.

Output

logicalRecordLockThreshold receives the number of explicit logical record locks allowed before implicit transactions begin.

physicalRecordLockThreshold receives the number of explicit physical record locks allowed before implicit transactions begin.

Return Values

0	Successful.	
-1	Unsuccessful.	One of the following error codes is placed in NWErrno:
	0xFE	Transaction Restart
	0xFD	Transaction Tracking Disabled
	0xFF	Lock Error
	0xFF	No Explicit Transaction Active

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

Description

This function and the NWTTSSetProcessThresholds function are useful for applications that change the implicit process thresholds and later want to restore them. For example, NWTTSGetProcessThresholds can query an application for the number of logical and physical record locks allowed before an implicit transaction begins. NWTTSSetProcessThresholds can then do one of the following:

- Turn off implicit transactions
- Set implicit thresholds for applications that always keep one or more records locked

The default threshold for logical and physical locks is 0. A threshold of 255 means there will be no implicit transactions allowed for that lock type.

The thresholds returned by this function are valid for the requesting application only. When the application terminates, the connection thresholds are restored.

Notes

Applications that intend to use only explicit transactions, but sometimes generate unnecessary implicit transactions, need to turn off all implicit transactions.

See Also

NWTTSSetProcessThresholds

NWTTSIsAvailable

This function verifies that the file server supports transaction tracking.

Synopsis

```
#include "nwapi.h"

NWBoolean_ts        ccode;
uint16              serverConnID;

ccode=NWTTSIsAvailable( serverConnID );
```

Input

serverConnID passes the file server connection ID.

Output

None.

Return Values

1 Transaction Tracking is available. (See ªDescriptionº below.)

0 Transaction Tracking is not available. (See ªDescriptionº below.) If another problem exists, one of the following error codes is placed in NWErrno:

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

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

Description

The version of NetWare that the file server is running determines the information that this call returns.

For NetWare v2.*x*, this call allows the application to know whether the server has TTS installed.

1	Indicates that TTS is installed.
0	Indicates that TTS is not installed.

However, the API does not indicate whether TTS is currently enabled. To ensure that TTS is enabled, NWTTSEnableTransactionTracking should be called

For NetWare v3.*x*, this call indicates whether TTS is enabled.

1	Indicates that TTS is enabled.
0	Indicates that TTS has been disabled.

See Also

NWTTSEnableTransactionTracking
NWTTSDisableTransactionTracking

NWTTSIsTransactionWritten

This function verifies whether a transaction has been written to disk.

Synopsis

#include "nwapi.h"

```
NWBoolean_ts        ccode;
uint16   serverConnID;
uint32   transactionReferenceNumber;

ccode=NWTTSIsTransactionWritten( serverConnID,
transactionReferenceNumber );
```

Input

serverConnID passes the file server connection ID.

transactionReferenceNumber passes the Transaction Reference number, obtained from the NWTTSEndTransaction function.

Output

None.

Return Values

1	Transaction written to disk
0	Transaction not written to disk, and an error code is placed in NWErrno:

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

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

Description

Before using NWTTSIsTransactionWritten, use NWTTSEndTransaction to obtain a valid transactionReferenceNumber. If NWTTSEndTransaction fails, do not use NWTTSIsTransactionWritten. When NWTTSEndTransaction fails, it returns an invalid transactionReferenceNumber. When NWTTSIsTransactionWritten is passed an invalid transactionReferenceNumber, it returns an invalid response.

Applications should not wait for transactions to be written to disk unless it is absolutely necessary. Because of the file server caching algorithms, it may be 3 to 5 seconds (or longer) before they are actually written.

Notes

Transactions are written to disk in the order in which they terminate.

See Also

NWTTSEndTransaction

NWTTSSetConnectionThresholds

This function informs NetWare of how many explicit physical and logical record locks to permit before invoking implicit transactions.

Synopsis

#include "nwapi.h"

```
int     ccode;
uint16  serverConnID;
uint8   logicalRecordLockThreshold;
uint8   physicalRecordLockThreshold;

ccode=NWTTSSetConnectionThresholds( serverConnID,
logicalRecordLockThreshold, physicalRecordLockThreshold );
```

Input

serverConnID passes the file server connection ID.

logicalRecordLockThreshold passes the number of logical record locks to allow before implicit transactions begin.

physicalRecordLockThreshold passes the number of physical record locks to allow before implicit transactions begin.

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

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

Description

This function and NWTTSGetConnectionThresholds are useful for applications that change the implicit application thresholds and later want to restore them. For example, NWTTSGetConnectionThresholds can obtain the current number of logical and physical locks and then NWTTSSetConnectionThresholds can do one of the following:

- Turn off implicit transactions. (Applications that use only explicit transactions, but sometimes generate unnecessary implicit transactions, need to turn off all implicit transactions.)
- Set implicit thresholds for applications that always keep one or more records locked.

Notes

The default threshold for logical and physical locks is 0. A threshold of 255 means no implicit transactions for that

lock type will be performed.

See Also

NWTTSGetConnectionThresholds

NWTTSSetControlFlags

This function enables or disables automatic record locking on writes to transactional files.

Synopsis

#include "nwapi.h"

```
int     ccode;
uint16  serverConnID;
uint8   TTSControlFlags;

ccode=NWTTSSetControlFlags( serverConnID, TTSControlFlags );
```

Input

serverConnID passes the file server connection ID.

TTSControlFlags passes the Transaction Tracking Control flags. (See "Description" below.)

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

See Appendix B for a complete listing of possible NetWare errors.

Description

Transaction tracking control flags are only valid for files flagged as TTS (transactional). These flags are defined as follows:

0x00	Automatic record locking is disabled
0x01	Automatic record locking is enabled

See Also

NWTTSGetControlFlags

NWTTSSetProcessThresholds

This function sets the number of logical and physical locks to perform before implicit locking begins.

Synopsis

#include ªnwapi.hº

```
int     ccode;
uint16  serverconnID;
uint8   logicalRecordLockThreshold;
uint8   physicalRecordLockThreshold;

ccode=NWTTSSetProcessThresholds ( serverConnID,
logicalRecordLockThreshold, physicalRecordLockThreshold );
```

Input

serverConnID passes the file server connection ID.

logicalRecordLockThreshold passes the number of logical record locks to allow before implicit transactions begin.

physicalRecordLockThreshold passes the number of physical record locks to allow before implicit transactions begin.

Output

None.

Return Values

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

0xFE	Transaction Restart
0xFD	Transaction Tracking Disabled
0xFF	Lock Error
0xFF	No Explicit Transaction Active

See Appendix B for a complete listing of possible NetWare errors.

Description

The thresholds set by this function are valid for the requesting application only. When the application terminates, the default workstation thresholds are restored.

This function is useful in either turning off implicit transactions or allowing applications that always keep one or more records locked to work. Applications that intend to use only explicit transactions, but sometimes generate unnecessary implicit transactions, can use this function to turn off all implicit transactions.

Notes

The default threshold for logical and physical locks is 0 unless this number has been changed using the NWTTSSetConnectionThresholds function. A threshold of 255 means no implicit transactions for that lock type are performed.

See Also

NWTTSGetProcessThresholds