Basic TxF Concepts - Win32 apps

By jwmsft

Archived: 2026-04-05 16:58:33 UTC

Read Isolation

Transactional NTFS (TxF) provides read-committed consistency.

A *transacted writer* refers to a transacted file handle opened with any permission that is not part of generic read access but is part of generic write access. A *transacted writer* views the most recent version of a file that includes all of the changes by the same transaction. There can be only one transacted writer per file. Non-transacted writers are always blocked by a transacted writer, even if the file is opened with shared-write permissions.

A *transacted reader* refers to a transacted file handle opened with any permission that is a part of generic read access but is not part of generic write access. A *transacted reader* views a committed version of the file that existed at the time the file handle was opened. The transacted reader is isolated from the effects of transacted writers. This provides a consistent view of the file only for the life of the file handle and blocks non-transacted writers.

Note

When a handle has been opened for modification with the **CreateFileTransacted** function, all subsequent opens of the file within that transaction whether read-only or not are converted by the system to be a transacted writer for the purposes of isolation and other transactional semantics. This means that subsequently, when a handle is opened for read-only access, the handle does not receive a view of the file prior to the start of the transaction; it receives the active transaction view of the file.

A non-transacted file handle does not see any changes made within a transaction until the transaction is committed. The non-transacted file handle receives an isolated view that is similar to a transacted reader, but unlike a transacted reader, it receives the file update when a transacted writer commits the transaction.

Isolation Levels

TxF provides read-committed isolation. This means that file updates are not seen outside the transaction. In addition, if a file is opened more than once while reading files within the transaction, you may see different results with each subsequent opening. Files that were available the first time you accessed them may not be available (because they were deleted), or vice versa.

Transactional Locking

Creating a transacted writer on a file *transactionally locks* the file. After a file is locked by a transaction, other file system operations external to the locking transaction that try to modify the transactionally locked file will fail with

either **ERROR_SHARING_VIOLATION** or **ERROR_TRANSACTIONAL_CONFLICT**.

The following table summarizes transactional locking.

File currently opened by

File open attempted by

Transacted

Non-Transacted

Reader

Reader/Writer

Reader

Reader/Writer

Transacted Reader

Yes

Yes

Yes

No2

Transacted Reader/Writer

Yes

No2

Yes

No2

Non-Transacted Reader

Yes

Yes

Yes

Yes

Non-Transacted Reader/Writer

No1

No1

Yes

Yes

1. Fails with **ERROR_TRANSACTIONAL_CONFLICT**
2. Fails with **ERROR_SHARING_VIOLATION**

If you open a named stream for a modification that is using a transaction, the entire file is required to be locked.

In addition to transactional locking, typical NTFS file-sharing rules apply.

You need to consider the following two file sharing modes in parallel:

- The transactional locking mode.
- Normal file-sharing modes.

Whichever mode is more restrictive is the one that applies.
