[MS-FSA]: File System Algorithms

This topic lists the Errata found in the MS-FSA document since it was last published. Since this topic is updated frequently, we recommend that you subscribe to these RSS or Atom feeds to receive update notifications.

Errata are subject to the same terms as the Open Specifications documentation referenced.

RSS
Atom

To view a PDF file of the errata for the previous versions of this document, see the following ERRATA Archives:

March 2, 2016 - Download

October 16, 2015 - Download

June 30, 2015 - Download

July 18, 2016 - Download

June 1, 2017 - Download

September 15, 2017 - Download

December 1, 2017 - Download

September 12, 2018 - Download

June 24, 2019 – Download

March 4, 2020 - Download

August 24, 2020 – Download

June 1, 2021 - Download

October 6, 2021 - Download

Errata below are for Protocol Document Version 36.0 – 2022/04/29.

Errata Published*	Description
2023/02/14	Section 2.1.1.4 Per Link, added link usage description, includes information duplication with reference to algorithm to retrieve the latest. Changed from: The object store MUST implement the following persistent attributes:<26> . . . Changed to: A Link structure connects a file name to a directory containing the file. Additionally, a Link duplicates certain information about the file (timestamps, sizes, etc.), that can be used to satisfy directory query operations (see section 2.1.5.6). Note that for performance reasons an object store MAY delay updating a Link's duplicated information following modifications to a file, resulting in directory queries returning stale information. Some file modifications require an

Errata Published*	Description
	immediate update of the duplicated information, which will be noted in this document by invoking the algorithm described in section 2.1.4.18. The object store MUST implement the following persistent attributes:<26> . . . Section 2.1.5.6.3 Directory Information Queries, added directory query usage description, includes information duplication with reference to algorithm to retrieve the latest. Changed from: This section describes how the object store processes directory queries for the following FileInformationClass values: . . . Changed to: Directory queries return requested information about files contained in the directory, based on the Link structures in Open.DirectoryList. Note that for performance reasons an object store MAY delay updating a Link's duplicated information following modifications to a file, resulting in directory queries returning stale information. Some file modifications require an immediate update of the duplicated information, which will be noted in this document by invoking the algorithm described in section 2.1.4.18. This section describes how the object store processes directory queries for the following FileInformationClass values: . . .
2022/08/09	In section 2.1.5.15.11, FileRenameInformation, revised renaming processing. Changed from: ▪ If *RemoveSourceLink* is TRUE: ▪ If **Open.File.FileType** is DirectoryFile ▪ *FilterMatch* = FILE_NOTIFY_CHANGE_DIR_NAME ▪ Else ▪ *FilterMatch* = FILE_NOTIFY_CHANGE_FILE_NAME ▪ EndIf ▪ If *MoveToNewDir* is TRUE or *AddTargetLink* is FALSE or *RemoveTargetLink* and *ExactCaseMatch* are TRUE: *Action* = FILE_ACTION_REMOVED ▪ Else ▪ *Action* = FILE_ACTION_REMOVED_OLD_NAME ▪ EndIf Changed to: ▪ If *RemoveSourceLink* is TRUE: ▪ If **Open.File.FileType** is DirectoryFile ▪ *FilterMatch* = FILE_NOTIFY_CHANGE_DIR_NAME ▪ Else ▪ *FilterMatch* = FILE_NOTIFY_CHANGE_FILE_NAME ▪ EndIf ▪ If *MoveToNewDir* is TRUE or *AddTargetLink* is FALSE or *RemoveTargetLink* and *ExactCaseMatch* are TRUE: *Action* = FILE_ACTION_REMOVED ▪ Else ▪ *Action* = FILE_ACTION_RENAMED_OLD_NAME ▪ EndIf
2022/07/26	Added revisions to section 2.1.5.2, Server Requests an Open of a Named Pipe. Please see the diff file.

Errata Published*	Description
2022/06/01	Added new section, 2.1.5.2, Server Requests an Open of a Named Pipe. Please see the diff file.
2022/06/01	In section 2.1.5.15.11, FileRenameInformation, added information about how NTFS prevents a race condition during renaming. Changed from: If Open.File contains open files as specified in section 2.1.4.2, the operation MUST be failed with STATUS_ACCESS_DENIED. Changed to: If Open.File contains open files as specified in section 2.1.4.2, the operation MUST be failed with STATUS_ACCESS_DENIED.<174> <174> On Windows NTFS, NTFS checks for open files beneath the directory being renamed (performs section 2.1.4.2), it records the count of open files. If there is a lease to break, NTFS requests the break and then goes back to the start of performing 2.1.5.15.11. NTFS waits for the lease break acknowledgment and restarts the rename operation. When NTFS performs section 2.1.4.2 again, it again records how many open files there are beneath the directory and compares that to the previous count. If the current count is greater than or equal to the previous count, NTFS fails the rename and prevents a possible race condition.
2022/05/27	In section 2.1.5.10.34, FSCTL_SET_INTEGRITY_INFORMATION_EX, updated list of applicable updates. Changed from: <127> Section 2.1.5.10.34: The FSCTL_SET_INTEGRITY_INFORMATION_EX operation is supported only by the ReFS file system v3.2 or higher (Windows 10 v1507 operating system or higher). FSCTL_SET_INTEGRITY_INFORMATION_EX is handled following the process in this section on systems updated with [MSKB-5014019], [MSKB-5014021], [MSKB-5014022], or [MSKB-5014023]. Changed to: <127> Section 2.1.5.10.34: The FSCTL_SET_INTEGRITY_INFORMATION_EX operation is supported only by the ReFS file system v3.2 or higher (Windows 10 v1507 operating system or higher). FSCTL_SET_INTEGRITY_INFORMATION_EX is handled following the process in this section on systems updated with [MSKB-5014019], [MSKB-5014021], [MSKB-5014022], [MSKB-5014023], [MSKB-5014701], [MSKB-5014702], or [MSKB-5014710].
2022/05/18	The following sections were changed. Please see the diff document for the details. In Section 2.1.1.3, Per File, updated a product behavior about how registry entries affect the handling of LastAccessTime: Changed from: <17> Section 2.1.1.3: In Windows Vista and subsequent, LastAccessTime updates are disabled by default in the ReFS and NTFS file systems. It is only updated when the file is closed. This behavior is controlled by the following registry key: HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate. A nonzero value means LastAccessTime updates are disabled. A value of zero means they are enabled. Changed to: <17> Section 2.1.1.3: In Windows Vista and subsequent, LastAccessTime updates are disabled by default in the ReFS and NTFS file systems. It is only updated when the file is closed. This behavior is controlled by the following registry values (respectively): HKLM\System\CurrentControlSet\Control\FileSystem\RefsDisableLastAccessUpdate, and HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate. A value of 1 means LastAccessTime updates are disabled. Any other value (or undefined) means they are enabled. In Windows 10 v1803 operating system and subsequent, NTFS has two registry values controlling LastAccessTime updates: HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate and HKLM\System\CurrentControlSet\Control\FileSystem\ NtfsLastAccessUpdatePolicyVolumeSizeThreshold. The NtfsDisableLastAccessUpdate value is now treated as a bitfield as follows:

<table>
<tr><th>Errata Published*</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0x00000001</td><td>Disable LastAccessTime updates.</td></tr>
<tr><td>0x00000002</td><td>System managed. Indicates that NTFS uses its own policy for updating LastAccessTime as follows:
On client systems, LastAccessTime updates are enabled if any of the following conditions are true:
● NtfsLastAccessUpdatePolicyVolumeSizeThreshold is 0.
● The size of the boot volume is <= NtfsLastAccessUpdatePolicyVolumeSizeThreshold in GB.
● NtfsLastAccessUpdatePolicyVolumeSizeThreshold is undefined and (prior to Windows 10 v2004) the size of the boot volume is <= 128GB.
On server systems, or client systems where the above conditions do not apply, LastAccessTime updates are always disabled.
At system startup, after evaluating the above policy, NTFS will set/clear flag 0x00000001 accordingly to reflect that LastAccessTime updates are disabled/enabled.</td></tr>
<tr><td>0x80000000</td><td>Flags initialized. Indicates NTFS recognizes flags other than 0x00000001. At system startup, if flag 0x80000000 is not set, the system will automatically set flag 0x80000000 and will additionally set flag 0x00000002 (becoming system managed) if flag 0x00000001 was set.</td></tr>
</table>
If the value of NtfsDisableLastAccessUpdate is controlled by group policy, then only flag 0x00000001 is recognized.

In Section 2.1.1.4, Per Link, updated a product behavior about how registry entries affect the handling of LastAccessTime:

Changed from:

<31> Section 2.1.1.4: In Windows Vista and subsequent LastAccessTime updates are disabled by default in the ReFS and NTFS file systems. It is only updated when the file is closed. This behavior is controlled by the following registry key: HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate. A nonzero value means LastAccessTime updates are disabled. A value of zero means they are enabled.

Changed to:

<31> Section 2.1.1.4: In Windows Vista and subsequent, LastAccessTime updates are disabled by default in the ReFS and NTFS file systems. It is updated only when the file is closed. This behavior is controlled by the following registry values (respectively): HKLM\System\CurrentControlSet\Control\FileSystem\RefsDisableLastAccessUpdate, and HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate. A value of 1 means LastAccessTime updates are disabled. Any other value (or undefined) means they are enabled.

In Windows 10 v1803 and subsequent, NTFS has two registry values controlling LastAccessTime updates: HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate and HKLM\System\CurrentControlSet\Control\FileSystem\ NtfsLastAccessUpdatePolicyVolumeSizeThreshold. The NtfsDisableLastAccessUpdate value is
</td></tr>
</table>

<table>
<tr><th>Errata Published*</th><th>Description</th></tr>
<tr><td></td><td>now treated as a bitfield as follows:

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0x00000001</td><td>Disable LastAccessTime updates.</td></tr>
<tr><td>0x00000002</td><td>System managed. Indicates that NTFS uses its own policy for updating LastAccessTime as follows:
On client systems, LastAccessTime updates are enabled if any of the following conditions are true:
• NtfsLastAccessUpdatePolicyVolumeSizeThreshold is 0.
• The size of the boot volume is less than or equal to NtfsLastAccessUpdatePolicyVolumeSizeThreshold in GB.
• NtfsLastAccessUpdatePolicyVolumeSizeThreshold is undefined and (prior to Windows 10 v2004) the size of the boot volume is <= 128GB.
On server systems, or client systems where the above conditions do not apply, LastAccessTime updates are always disabled.
At system startup, after evaluating the above policy, NTFS will set/clear flag 0x00000001 accordingly to reflect that LastAccessTime updates are disabled/enabled.</td></tr>
<tr><td>0x80000000</td><td>Flags initialized. Indicates NTFS recognizes flags other than 0x00000001. At system startup, if flag 0x80000000 is not set, the system will automatically set flag 0x80000000 and will additionally set flag 0x00000002 (becoming system managed) if flag 0x00000001 was set.</td></tr>
</table>

If the value of NtfsDisableLastAccessUpdate is controlled by group policy, then only flag 0x00000001 is recognized.</td></tr>
<tr><td>2022/05/02</td><td>In Section 2.1.5.9.34, FSCTL_SET_INTEGRITY_INFORMATION_EX, updated processing rules for system versions.

Changed from:
The server provides:<127>

<127> Section 2.1.5.9.34: The FSCTL_SET_INTEGRITY_INFORMATION_EX operation is supported only by the ReFS file system v3.2 or higher (Windows 10 v1507 operating system or higher).

Changed to:

The server provides:<127>

<127> Section 2.1.5.9.34: The FSCTL_SET_INTEGRITY_INFORMATION_EX operation is supported only by the ReFS file system v3.2 or higher (Windows 10 v1507 operating system or higher). FSCTL_SET_INTEGRITY_INFORMATION_EX is handled following the process in this section on systems updated with [MSKB-5014019], [MSKB-5014021], [MSKB-5014022], or [MSKB-5014023].</td></tr>
</table>