5 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.1: Of the standard Windows file systems, only the UDFS file system supports Software Defect Management.

<2> Section 2.1.1.1: NTFS uses a default cluster size of 4 KB, a maximum cluster size of 64 KB, and a minimum cluster size of 512 bytes. ReFS uses a default cluster size of 64 KB, a maximum cluster size of 128k, and a minimum cluster size of 4 KB. ReFS is supported only on Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2.

<3> Section 2.1.1.1: For AMD64, x86, and ARM systems, this value is 4 KB. For ia64 systems, this value is 8 KB.

<4> Section 2.1.1.1: In NTFS, the CompressionUnitSize is 64 KB for encrypted files, 64 KB for sparse files, and the lesser of 64 KB or (16 * **ClusterSize**) for compressed files. Other file systems do not implement this field.

<5> Section 2.1.1.1: In NTFS, the CompressedChunkSize is 4 KB. Other Windows file systems do not implement this field.

<6> Section 2.1.1.1: Only ReFS supports integrity.

<7> Section 2.1.1.1: Only NTFS supports quotas.

<8> Section 2.1.1.1: This field is present for compatibility with the file level FileObjectIdInformation structure ([MS-FSCC] section 2.4.28). These fields are not currently used by Windows and always contain zeroes.

<9> Section 2.1.1.1: The USN journal is supported on ReFS all versions and NTFS version 3.0 volumes or greater. The USN journal is active by default on Windows client SKUs starting with Windows Vista and later. The USN journal is not active by default on Windows Server SKUs.

<10> Section 2.1.1.1: For Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the maximum file size of a file on an NTFS volume is the smaller of (2^{32} – 1) * cluster size, and 16 terabytes (TB). For Windows 8 and Windows Server 2012, the maximum file size of a file on an NTFS volume is (2^{32} – 1) * cluster size. For Windows 8.1 and Windows Server 2012 R2, the maximum file size of a file on an NTFS volume is ((2^{32} * cluster size) – 64K). For example, if the cluster size is 512 bytes, the maximum file size is 2 TB.

<11> Section 2.1.1.2: ReFS does not implement the TunnelCache.

<12> Section 2.1.1.3: ReFS and exFAT do not implement **ShortNames**.

<13> Section 2.1.1.3: The following table defines the support of file time stamps across various Windows file systems. More information can be found in section 6 of the File System Behavior Overview document [FSBO].

Timestamp	ReFS	NTFS	FAT	EXFAT	UDFS
CreationTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 10 millisecond granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity
LastAccessTime	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in local time 1 day granularity	Stored in UTC if available, else in local time 2 second granularity	Stored in UTC if available, else in local time 1 microsecond granularity
ChangeTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Not Supported	Not Supported	Stored in UTC if available, else in local time 1 microsecond granularity
LastWriteTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 2 second granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity

<14> Section 2.1.1.3: The following table defines the support of file time stamps across various Windows file systems. More information can be found in section 6 of the File System Behavior Overview document [FSBO].

Timestamp	ReFS	NTFS	FAT	EXFAT	UDFS

CreationTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 10 millisecond granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity
LastAccessTime	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in local time 1 day granularity	Stored in UTC if available, else in local time 2 second granularity	Stored in UTC if available, else in local time 1 microsecond granularity
ChangeTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Not Supported	Not Supported	Stored in UTC if available, else in local time 1 microsecond granularity
LastWriteTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 2 second granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity

<15> Section 2.1.1.3: The following table defines the support of file time stamps across various Windows file systems. More information can be found in section 6 of the File System Behavior Overview document [FSBO].

Timestamp	ReFS	NTFS	FAT	EXFAT	UDFS
CreationTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 10 millisecond granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity
LastAccessTime	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in local time 1 day granularity	Stored in UTC if available, else in local time 2 second granularity	Stored in UTC if available, else in local time 1 microsecond granularity
ChangeTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Not Supported	Not Supported	Stored in UTC if available, else in local time 1 microsecond granularity
LastWriteTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 2 second granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity

<16> Section 2.1.1.3: In Windows Vista/Windows Server 2008 and later, 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.

<17> Section 2.1.1.3: The following table defines the support of file time stamps across various Windows file systems. More information can be found in section 6 of the File System Behavior Overview document [FSBO].

Timestamp	ReFS	NTFS	FAT	EXFAT	UDFS
CreationTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 10 millisecond granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity
LastAccessTime	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in UTC 100 nanosecond granularity Updated at 60 minute granularity	Stored in local time 1 day granularity	Stored in UTC if available, else in local time 2 second granularity	Stored in UTC if available, else in local time 1 microsecond granularity
ChangeTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Not Supported	Not Supported	Stored in UTC if available, else in local time 1 microsecond granularity
LastWriteTime	Stored in UTC 100 nanosecond granularity	Stored in UTC 100 nanosecond granularity	Stored in local time 2 second granularity	Stored in UTC if available, else in local time 10 millisecond granularity	Stored in UTC if available, else in local time 1 microsecond granularity

<18> Section 2.1.1.3: Only NTFS implements EAs.

<19> Section 2.1.1.3: Only NTFS implements EAs.

<20> Section 2.1.1.3: Only NTFS implements object IDs.

<21> Section 2.1.1.3: Only NTFS implements object IDs.

<22> Section 2.1.1.3: Only NTFS and UDFS implement named streams.

<23> Section 2.1.1.3: ReFS and exFAT do not implement **ShortNames**.

<24> Section 2.1.1.3: Only NTFS implements encryption.

<25> Section 2.1.1.4: For ReFS, there will always be exactly one link per file or directory.

<26> Section 2.1.1.4: On ReFS or exFAT, this field MUST be empty.

<27> Section 2.1.1.5: Only NTFS supports compression.

<28> Section 2.1.1.5: Only ReFS supports integrity.

<29> Section 2.1.1.5: Only ReFS supports integrity.

<30> Section 2.1.1.5: Only NTFS and UDFS support sparse files.

<31> Section 2.1.1.5: Only NTFS supports encryption.

<32> Section 2.1.1.6: Only NTFS implements EAs.

<33> Section 2.1.4.11: NTFS sets *RecordLength* to *BlockAlign***(***FieldOffset***(**USN_RECORD_V2.FileName**)** + *FileNameLength*, 8**)**. ReFS sets *RecordLength* to *BlockAlign***(***FieldOffset***(**USN_RECORD_V3.FileName**)** + *FileNameLength*, 8**)**.

<34> Section 2.1.5.1: NTFS and ReFS recognize the following stream type names:

- "$STANDARD_INFORMATION"
- "$ATTRIBUTE_LIST"
- "$FILE_NAME"
- "$OBJECT_ID"
- "$SECURITY_DESCRIPTOR"
- "$VOLUME_NAME"
- "$VOLUME_INFORMATION"
- "$DATA"
- "$INDEX_ROOT"
- "$INDEX_ALLOCATION"
- "$BITMAP"
- "$REPARSE_POINT"
- "$EA_INFORMATION"
- "$EA"
- "$LOGGED_UTILITY_STREAM"

Other Windows file systems do not recognize any stream type names.

<35> Section 2.1.5.1.1: For the NTFS file system, the **FileId128** consists of a 48-bit index into the MFT (the low 48 bits) and a 16-bit sequence number (the next higher 16 bits), with the high 64 bits unused and always equal to 0. For the ReFS file system, the **FileId128** consists of a 64-bit index uniquely identifying the file's parent directory on the volume (the low 64 bits) and a 64-bit index uniquely identifying the file within that directory (the high 64 bits).

<36> Section 2.1.5.1.1: For the NTFS file system this is the index and sequence number portions (low 64 bits) of the **FileId128**. The ReFS file system maps a subset of the possible **FileId128** values to **FileId64** values using a reversible algorithm; for values outside of this subset, ReFS sets the **FileId64** to -1.

<37> Section 2.1.5.1.1: For the NTFS file system, this is the index portion (low 48 bits) of the **FileId128**. The ReFS file system does not implement this field.

<38> Section 2.1.5.1.1: Only ReFS supports FILE_ATTRIBUTE_INTEGRITY_STREAM.

<39> Section 2.1.5.1.1: Only NTFS and ReFS support FILE_ATTRIBUTE_NO_SCRUB_DATA.

<40> Section 2.1.5.1.1: Only NTFS and UDFS implement named streams.

<41> Section 2.1.5.1.2: Windows 2000, Windows XP, Windows Server 2003, and Windows Vista, treat the FILE_DISALLOW_EXCLUSIVE option as always being FALSE.

<42> Section 2.1.5.5.1: This is implemented only by the NTFS file system.

<43> Section 2.1.5.5.1: This directory is only available on NTFS volumes formatted to NTFS version 3.0 or late.

<44> Section 2.1.5.5.1: "*" is treated as 0x0000002A during the search, and it gives the practical behavior of a wildcard since an ObjectId starts with a much larger value. Similarly, "?" is treated as 0x0000003F and so practically it behaves like "*".

<45> Section 2.1.5.5.2: This is implemented only by the NTFS file system.

<46> Section 2.1.5.5.2: This directory is only available on NTFS volumes formatted to NTFS version 3.0 or later.

<47> Section 2.1.5.5.3.1: For ReFS, this value MUST be zero.

<48> Section 2.1.5.5.3.3: For ReFS, this value MUST be zero.

<49> Section 2.1.5.5.3.4: For ReFS, this value MUST be zero.

<50> Section 2.1.5.5.3.5: For ReFS, this value MUST be zero.

<51> Section 2.1.5.6: This is only implemented by the NTFS file system. Other file systems return STATUS_SUCCESS and perform no other action.

<52> Section 2.1.5.9.1: This is only implemented by the NTFS file system.

<53> Section 2.1.5.9.1: If the generated ObjectId collides with existing ObjectIds on the volume, Windows retries up to 16 times before failing the operation with STATUS_DUPLICATE_NAME.

<54> Section 2.1.5.9.1: The file system only updates **LastChangeTime** if no user has explicitly set **LastChangeTime**. The NTFS and ReFS file systems defer setting **LastChangeTime** until the handle is closed.

<55> Section 2.1.5.9.2: This is only implemented by the NTFS file system.

<56> Section 2.1.5.9.2: The file system only updates **LastChangeTime** if no user has explicitly set **LastChangeTime**. The NTFS and ReFS file systems defer setting **LastChangeTime** until the handle is closed.

<57> Section 2.1.5.9.3: This is only implemented by the NTFS file system.

<58> Section 2.1.5.9.3: The file system only updates **LastChangeTime** if no user has explicitly set **LastChangeTime**. The NTFS and ReFS file systems defer setting **LastChangeTime** until the handle is closed.

<59> Section 2.1.5.9.4: If the Open is a directory on a Cluster Shared Volume File System (CSVFS), the operation MUST be failed with STATUS_NOT_IMPLEMENTED.

<60> Section 2.1.5.9.5: This is only implemented by the ReFS, NTFS, FAT, and exFAT file systems.

<61> Section 2.1.5.9.5: The NTFS file system sets an NTFS_STATISTICS structure as specified in [MS-FSCC] section 2.3.8.2. The FAT file system sets a FAT_STATISTICS structure as specified in [MS-FSCC] section 2.3.8.3. The EXFAT file system sets a EXFAT_STATISTICS structure as specified in [MS-FSCC] section 2.3.8.4.

<62> Section 2.1.5.9.6: This is only implemented by the NTFS file system.

<63> Section 2.1.5.9.6: Some file systems have more efficient mechanisms to obtain a list of files. For instance, NTFS iterates through all base file records of the MFT.

<64> Section 2.1.5.9.7: This is only implemented by the NTFS and ReFS file systems.

<65> Section 2.1.5.9.8: This operation is only implemented by the ReFS file system.

<66> Section 2.1.5.9.9: This is only implemented by the NTFS file system.

<67> Section 2.1.5.9.9: Several of the fields being set in this section are specific to how the NTFS file system is implemented and are not defined in the Object Stores Abstract Data Model.

<68> Section 2.1.5.9.11: This is only implemented by the NTFS file system.

<69> Section 2.1.5.9.12: This is only implemented by the ReFS and NTFS file systems.

<70> Section 2.1.5.9.17: This is implemented only by the NTFS file system.

<71> Section 2.1.5.9.18: This is implemented only by the NTFS file system.

<72> Section 2.1.5.9.19: This is only implemented by the ReFS and NTFS file systems.

<73> Section 2.1.5.9.20: Support for this FSCTL is only implemented in the FAT file system. The data returned by this FSCTL is incomplete and incorrect on FAT32, and it is unsupported on all other file systems, as specified in [MS-FSCC] section 2.3.39.

<74> Section 2.1.5.9.22: This is only implemented by the UDFS file system.

<75> Section 2.1.5.9.23: This is only implemented by the UDFS file system.

<76> Section 2.1.5.9.24: This is only implemented by the ReFS and NTFS file systems.

<77> Section 2.1.5.9.24: In Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012, NTFS uses a *MaxMajorVersionSupported* value of 2.

<78> Section 2.1.5.9.24: In Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7 and Windows Server 2008 R2, NTFS ignores the input buffer completely; all requests are treated as having an **InputBufferSize** of 0.

<79> Section 2.1.5.9.24: In Windows 8 and Windows Server 2012, the operation MUST be failed with STATUS_NOT_IMPLEMENTED.

<80> Section 2.1.5.9.25: This file system request is handled by the optional hierarchical storage management (HSM) file system filter. This filter has been deprecated as of Windows Server 2008 and is a server-only feature.

<81> Section 2.1.5.9.26: If the Open is a directory on a Cluster Shared Volume File System (CSVFS), the operation MUST be failed with STATUS_NOT_IMPLEMENTED.

<82> Section 2.1.5.9.26: This method is fully supported with NTFS, but for ReFS, it is only supported and returns STATUS_SUCCESS when **CompressionState** is set to COMPRESSION_FORMAT_NONE. The method fails with STATUS_NOT_SUPPORTED for any other value of **CompressionState**.

<83> Section 2.1.5.9.26: NTFS File Compression can be disabled globally on a system by setting the registry key HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsDisableCompression to 1 and then rebooting the system to have the change take effect. Compression can be re-enabled by setting this key to zero and rebooting the system.

<84> Section 2.1.5.9.27: This is only implemented by the UDFS file system on media types that require software defect management.

<85> Section 2.1.5.9.28: This is only implemented by the NTFS file system.

<86> Section 2.1.5.9.29: Only ReFS supports integrity.

<87> Section 2.1.5.9.29: If the Open is a directory on a Cluster Shared Volume File System (CSVFS), the operation MUST be failed with STATUS_NOT_IMPLEMENTED.

<88> Section 2.1.5.9.29: This is implemented only by the ReFS file system.

<89> Section 2.1.5.9.30: This is only implemented by the NTFS file system.

<90> Section 2.1.5.9.30: The file system only updates **LastChangeTime** if no user has explicitly set **LastChangeTime**. The NTFS and ReFS file systems defer setting **LastChangeTime** until the handle is closed.

<91> Section 2.1.5.9.31: This is only implemented by the NTFS file system.

<92> Section 2.1.5.9.31: The file system only updates LastChangeTime if no user has explicitly set LastChangeTime. The NTFS and ReFS file systems defer setting the LastChangeTime until the handle is closed.

<93> Section 2.1.5.9.32: This is only implemented by the ReFS and NTFS file systems.

<94> Section 2.1.5.9.32: The file system only updates LastChangeTime if no user has explicitly set LastChangeTime. The NTFS and ReFS file systems defer setting the LastChangeTime until the handle is closed.

<95> Section 2.1.5.9.33: WinPE stands for the Windows Preinstallation Environment. For more information please see [MSFT-WinPW].

<96> Section 2.1.5.9.34: If the Open is a directory on a Cluster Shared Volume File System (CSVFS), the operation MUST be failed with STATUS_NOT_IMPLEMENTED.

<97> Section 2.1.5.9.34: This is only implemented by the NTFS file system and by the ReFS file system on non-integrity streams. In Windows 8.1 and Windows Server 2012 R2, ReFS supports this for both conventional and integrity streams.

<98> Section 2.1.5.9.35: If the Open is a directory on a Cluster Shared Volume File System (CSVFS), the operation MUST be failed with STATUS_NOT_IMPLEMENTED.

<99> Section 2.1.5.9.35: This is only implemented by the NTFS file system and by the ReFS file system on non-integrity streams. In Windows 8.1 and Windows Server 2012 R2, ReFS supports this for both conventional and integrity streams.

<100> Section 2.1.5.9.36: This is only implemented by the NTFS file system.

<101> Section 2.1.5.9.37: [SIS] (Single Instance Storage) is an optional feature available in the following versions of Windows Server: Windows Storage Server 2003 R2, Standard Edition, Windows Storage Server 2008, and Windows Storage Server 2008 R2. [SIS] is not supported directly by any of the Windows file systems but is implemented as a file system filter. Please refer to the following article for detailed information about [SIS].

<102> Section 2.1.5.9.37: This is implemented only by the NTFS file system.

<103> Section 2.1.5.9.37: In the Windows environment file system are implemented in kernel mode. If a NULL security context is specified and the originator of the operation is running in kernel mode, a built-in SYSTEM security context is used that grants all access.

<104> Section 2.1.5.9.37: In the Windows environment file system are implemented in kernel mode. If a NULL security context is specified and the originator of the operation is running in kernel mode, a built-in SYSTEM security context is used that grants all access.

<105> Section 2.1.5.9.37: In the Windows environment this is done by creating a new file in what is known as the "SIS Common Store". Reparse points are attached to any file controlled by [SIS] that contains information on how to access the Common Store file that contains the data for this file. Please see the following article about [SIS] for details on how this is

implemented.

<106> Section 2.1.5.9.38: This is only implemented by the NTFS file system.

<107> Section 2.1.5.11.5: Only ReFS supports integrity.

<108> Section 2.1.5.11.5: Only ReFS supports integrity.

<109> Section 2.1.5.11.6: Only ReFS supports integrity.

<110> Section 2.1.5.11.6: Only ReFS supports integrity.

<111> Section 2.1.5.11.10: Only NTFS implements EAs.

<112> Section 2.1.5.11.12: This operation is only supported by the NTFS file system.

<113> Section 2.1.5.11.21: Available only in ReFS.

<114> Section 2.1.5.11.21: Available only in ReFS.

<115> Section 2.1.5.11.23: If **Open.Mode** contains neither FILE_SYNCHRONOUS_IO_ALERT nor FILE_SYNCHRONOUS_IO_NONALERT, this operation does not return meaningful information in **OutputBuffer.CurrentByteOffset**, because **Open.CurrentByteOffset** is not maintained for any **Open** that does not have either of those flags set.

<116> Section 2.1.5.11.27: This algorithm is only implemented by NTFS and ReFS. The FAT, EXFAT, CDFS, and UDFS file systems always return 1.

<117> Section 2.1.5.12.5: The following table defines what FileSystemAttributes flags, as defined in [MS-FSCC] section 2.5.1, are set by various Windows file systems and why they are set:

	ReFS	NTFS	FAT	EXFAT	UDFS	CDFS
FILE_SUPPORTS_USN_JOURNAL 0x02000000	Always Set	Set if 3.0 format or higher volume				
FILE_SUPPORTS_OPEN_BY_FILE_ID 0x01000000	Always Set	Always Set			Set if volume mounted read-only	Always Set
FILE_SUPPORTS_EXTENDED_ATTRIBUTES 0x00800000		Always Set				
FILE_SUPPORTS_HARD_LINKS 0x00400000		Always Set			Always Set	
FILE_SUPPORTS_TRANSACTIONS 0x00200000		Set if 3.0 format or higher volume				
FILE_SEQUENTIAL_WRITE_ONCE 0x00100000					Set if volume not mounted read-only	

FILE_READ_ONLY_VOLUME 0x00080000	Set if volume mounted read-only	Set if volume mounted read-only	Set if volume mounted read-only	Set if volume mounted read-only	Set if volume mounted read-only	Always Set
FILE_NAMED_STREAMS 0x00040000		Always Set			Set if 2.0 format or higher	
FILE_SUPPORTS_ENCRYPTION 0x00020000		Set if 3.0 format or higher volume and encryption is enabled on the system				
FILE_SUPPORTS_OBJECT_IDS 0x00010000		Set if 3.0 format or higher volume				
FILE_VOLUME_IS_COMPRESSED 0x00008000						
FILE_SUPPORTS_REMOTE_STORAGE 0x00000100						
FILE_SUPPORTS_REPARSE_POINTS 0x00000080	Always Set	Set if 3.0 format or higher volume				
FILE_SUPPORTS_SPARSE_FILES 0x00000040		Set if 3.0 format or higher volume				
FILE_VOLUME_QUOTAS 0x00000020		Set if 3.0 format or higher volume				
FILE_FILE_COMPRESSION 0x00000010		Set if volumecluster size is 4K or less				
FILE_PERSISTENT_ACLS 0x00000008	Always Set	Always Set				
FILE_UNICODE_ON_DISK 0x00000004	Always Set	Always Set	Always Set	Always Set	Always Set	Set if Joliet Format
FILE_CASE_PRESERVED_NAMES 0x00000002	Always Set	Always Set	Always Set	Always Set	Always Set	
FILE_CASE_SENSITIVE_SEARCH 0x00000001	Always Set	Always Set			Always Set	Always Set

<118> Section 2.1.5.12.5: The following table defines the MaximumComponentNameLength, as defined in [MS-FSCC] section 2.5.1, that is set by each file system:

	ReFS	NTFS	FAT	EXFAT	UDFS	CDFS
MaximumComponentNameLength Value	255	255	255	255	254	110 if Joliet Format 221 otherwise

<119> Section 2.1.5.12.6: This is implemented only by the NTFS file system.

<120> Section 2.1.5.12.8: ReFS does not implement object IDs.

<121> Section 2.1.5.12.8: This is implemented only by the NTFS file system.

<122> Section 2.1.5.14.1: The following table describes the maximum file size supported by various Windows File Systems.

	ReFS	NTFS	FAT	EXFAT	UDFS	CDFS
MaximumFileSize	((2^32)-1) * ClusterSize	16 TB for Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 (((2^32)-1) * ClusterSize) for Windows 8 and Windows Server 2012 (((2^32) * ClusterSize) - 64K) for Windows 8.1 and Windows Server 2012 R2 The physical format will support 16 exabytes.	4 GB	16 exabytes	8 TB	8 TB

<123> Section 2.1.5.14.4: The following table describes the maximum file size supported by various Windows File Systems.

	ReFS	NTFS	FAT	EXFAT	UDFS	CDFS
MaximumFileSize	((2^32)-1) * ClusterSize	16 TB for Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 (((2^32)-1) * ClusterSize) for Windows 8 and Windows Server 2012 (((2^32) * ClusterSize) - 64K) for Windows 8.1 and Windows Server 2012 R2 The physical format will support 16 exabytes.	4 GB	16 exabytes	8 TB	8 TB

<124> Section 2.1.5.14.5: Only NTFS implements EAs.

<125> Section 2.1.5.14.6: In Windows, both the NTFS and UDFS file systems support hard links. UDFS support of hard links was added in Windows Vista and Windows Server 2008. ReFS does not support hard links.

<126> Section 2.1.5.14.9: If **Open.Mode** contains neither FILE_SYNCHRONOUS_IO_ALERT nor FILE_SYNCHRONOUS_IO_NONALERT, this operation does not have any meaningful effect, because **Open.CurrentByteOffset** is not used for any **Open** that does not have either of those flags set.

<127> Section 2.1.5.14.11: The file system only updates **LastChangeTime** if no user has explicitly set **LastChangeTime**. The NTFS and ReFS file systems defer setting **LastChangeTime** until the handle is closed.

<128> Section 2.1.5.14.13: ReFS does not implement short names.

<129> Section 2.1.5.14.14: ValidDataLength is an internal implementation detail of the NTFS file system and the ReFS file system. It is not a notion that exists in other Windows file systems. ValidDataLength, as defined by NTFS and ReFS, refers to a high-watermark in the file that is considered to be initialized data by a user writing in the region or by the file system writing zeros. Any reads within that value are required to return data from the persistent store. Any reads beyond that value

are required to return zeros. There is no API to query ValidDataLength, and the API to set ValidDataLength only allows the value to increase from the existing value.

<130> Section 2.1.5.15.6: This is implemented only by the NTFS file system.

<131> Section 2.1.5.15.8: Only NTFS implements object IDs.

<132> Section 2.1.5.15.8: This is only implemented by the NTFS file system.

<133> Section 2.1.5.16: The file system only updates **LastChangeTime** if no user has explicitly set **LastChangeTime**. The NTFS and ReFS file systems defer setting **LastChangeTime** until the handle is closed.

<134> Section 2.1.5.19: In Windows file systems, operations are only cancelable if they are blocked and put on a wait queue of some kind. Operations that are actively being processed are not cancelable.

<135> Section 2.1.5.20: The name of the quota file in the Windows environment is:

$Extend\$Quota:$Q:$INDEX_ALLOCATION

<136> Section 2.1.5.20: This operation is implemented only by the NTFS file system.

<137> Section 2.1.5.21: The name of the quota file in the Windows environment is:

$Extend\$Quota:$Q:$INDEX_ALLOCATION

<138> Section 2.1.5.21: This operation is only implemented by the NTFS file system.