[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

Errata below are for Protocol Document Version V27.0 – 2018/03/16.

Errata Published*	Description
2018/09/03	In Section 2.1.5.9.5, FSCTL_DUPLICATE_EXTENTS_TO_FILE_EX, the following has been added to the Pseudocode: • If InputBuffer.StructureSize is not equal to sizeof(DUPLICATE_EXTENTS_DATA_EX), the operation MUST be failed with STATUS_NOT_SUPPORTED.
2018/08/20	In Section 2.1.5.6, Server Requests Flushing Cached Data, the content has been changed from: The server provides: • Open: An Open of a DataFile or DirectoryFile for which it is to flush cached data. On completion, the object store MUST return: • Status: An NTSTATUS code that specifies the result. The object store MUST flush all persistent attributes for Open.File to stable storage. In addition: • If Open.File.Volume.IsReadOnly is TRUE, the operation MUST be failed with STATUS_MEDIA_WRITE_PROTECTED. • The operation MUST be failed with the status code returned from the underlying physical storage. The operation flushes all eligible objects; however, only the first failure encountered is returned. • The operation ensures that the directory structure is persisted to stable storage.<62> Pseudocode for the operation is as follows: • If Open.FileType is DirectoryFile: • CurrentDirectory = Open.DirectoryFile • Flush CurrentDirectory • While CurrentDirectory != CurrentDirectory.Volume.RootDirectory: • Set CurrentLink to the head of CurrentDirectory.LinkList, which is the only link because directories cannot have hard links. • CurrentDirectory = CurrentLink.ParentFile • Flush CurrentDirectory • EndWhile • EndIf • Flush all open objects on the volume. • If Open.File is equal to Open.File.Volume.RootDirectory: • For each OpenFile in Open.File.Volume.OpenFileList: • Flush OpenFile • EndFor • EndIf

Errata Published*	Description
	Changed to: The server provides: • Open: An Open of a DataFile or DirectoryFile for which it is to flush cached data. On completion, the object store MUST return: • Status: An NTSTATUS code that specifies the result. The object store MUST flush all persistent attributes for Open.File to stable storage. In addition: • If Open.File.Volume.IsReadOnly is TRUE, the operation MUST be failed with STATUS_MEDIA_WRITE_PROTECTED. • The operation MUST be failed with the status code returned from the underlying physical storage. The operation flushes all eligible objects; however, only the first failure encountered is returned. • The operation ensures that the directory structure is persisted to stable storage.<62> Pseudocode for the operation is as follows: • If Open.Stream.StreamType is DataStream: • Flush cached data of Open.File • Flush file system metadata associated with Open.File. • Else if Open.Stream.StreamType is DirectoryStream: • Flush file system metadata associated with Open.File • Else if Open.File is equal to Open.File.Volume.RootDirectory: • For each OpenFile in Open.File.Volume.OpenFileList: • Flush OpenFile • Flush file system metadata associated with OpenFile • EndFor • EndIf Flush the underlying physical storage.
2018/07/16	In Section 2.1.5.1, Server Requests an Open of a File Phase 7, changed from: • Phase 7 -- Type of file to open: • The object store MUST use the following algorithm to determine which type of file is being opened: • Set FileTypeToOpen to empty. • If RootOpen.File.Volume.IsPhysicalRoot is TRUE, then set FileTypeToOpen to ViewIndexFile under any of the following conditions: • If RootOpen.File.Volume.IsObjectIDsSupported is TRUE, BuildRelativeName(Open.Link, Open.File.Volume.RootDirectory) is equal to "\$Extend\$ObjId", StreamNameToOpen is equal to "$O", and StreamTypeNameToOpen is equal to "$INDEX_ALLOCATION" (using case-insensitive string comparisons). • If RootOpen.File.Volume.IsQuotasSupported is TRUE, BuildRelativeName(Open.Link, Open.File.Volume.RootDirectory) is equal to "\$Extend\$Quota", StreamNameToOpen is equal to "$O" or "$Q", and StreamTypeNameToOpen is equal to "$INDEX_ALLOCATION" (using case-insensitive string comparisons). • If RootOpen.File.Volume.IsReparsePointsSupported is TRUE, BuildRelativeName(Open.Link, Open.File.Volume.RootDirectory) is equal to "\$Extend\$Reparse", StreamNameToOpen is equal to "$R", and StreamTypeNameToOpen is equal to "$INDEX_ALLOCATION" (using case-insensitive string comparisons). • EndIf

Errata Published*	Description
	• // Note that when FileTypeToOpen is ViewIndexFile, the file always exists in the object store and • // Open.File.FileType is ViewIndexFile. • If FileTypeToOpen is empty: • If StreamTypeNameToOpen is "$INDEX_ALLOCATION" and StreamNameToOpen has a value other than an empty stream or "$I30", the operation SHOULD<44> be failed with STATUS_INVALID_PARAMETER. • If CreateOptions.FILE_DIRECTORY_FILE is TRUE then FileTypeToOpen = DirectoryFile. • Else if CreateOptions.FILE_NON_DIRECTORY_FILE is TRUE then FileTypeToOpen = DataFile. • Else if StreamTypeNameToOpen is "$INDEX_ALLOCATION" then FileTypeToOpen = DirectoryFile. • Else if StreamTypeNameToOpen is "$DATA" then FileTypeToOpen = DataFile. • Else if Open.File is not NULL and Open.File.FileType is DirectoryFile, then FileTypeToOpen = DirectoryFile. • Else if PathName contains a trailing backslash then FileTypeToOpen = DirectoryFile. • Else FileTypeToOpen = DataFile. • EndIf • If FileTypeToOpen is DirectoryFile and Open.File is not NULL and Open.File.FileType is not DirectoryFile: • If CreateDisposition == FILE_CREATE then the operation MUST be failed with STATUS_OBJECT_NAME_COLLISION, else the operation MUST be failed with STATUS_NOT_A_DIRECTORY. • EndIf • If FileTypeToOpen is DataFile and StreamNameToOpen is empty and Open.File is not NULL and Open.File.FileType is DirectoryFile, the operation MUST be failed with STATUS_FILE_IS_A_DIRECTORY. Changed to: • Phase 7 -- Type of stream to open: • The object store MUST use the following algorithm to determine which type of stream is being opened: • Set StreamTypeToOpen to empty. • If RootOpen.File.Volume.IsPhysicalRoot is TRUE, then set StreamTypeToOpen to ViewIndexStream under any of the following conditions: • If RootOpen.File.Volume.IsObjectIDsSupported is TRUE, BuildRelativeName(Open.Link, Open.File.Volume.RootDirectory) is equal to "\$Extend\$ObjId", StreamNameToOpen is equal to "$O", and StreamTypeNameToOpen is equal to "$INDEX_ALLOCATION" (using case-insensitive string comparisons). • If RootOpen.File.Volume.IsQuotasSupported is TRUE, BuildRelativeName(Open.Link, Open.File.Volume.RootDirectory) is equal to "\$Extend\$Quota", StreamNameToOpen is equal to "$O" or "$Q", and StreamTypeNameToOpen is equal to "$INDEX_ALLOCATION" (using case-insensitive string comparisons). • If RootOpen.File.Volume.IsReparsePointsSupported is TRUE, BuildRelativeName(Open.Link, Open.File.Volume.RootDirectory) is equal to "\$Extend\$Reparse", StreamNameToOpen is equal to "$R", and StreamTypeNameToOpen is equal to "$INDEX_ALLOCATION" (using case-insensitive string comparisons). • EndIf

Errata Published*	Description
	• // Note that when StreamTypeToOpen is ViewIndexStream, the file always exists in the object store and • // Open.File.FileType is ViewIndexFile. • If StreamTypeToOpen is empty: • If StreamTypeNameToOpen is "$INDEX_ALLOCATION": • If StreamNameToOpen has a value other than an empty string or "$I30", the operation SHOULD<44> be failed with STATUS_INVALID_PARAMETER. • Else if StreamTypeNameToOpen is not "$DATA" and not empty: • If CreateDisposition is one of FILE_SUPERSEDE, FILE_OVERWRITE, or FILE_OVERWRITE_IF, then the operation MUST be failed with STATUS_ACCESS_DENIED. • EndIf • If CreateOptions.FILE_DIRECTORY_FILE is TRUE then StreamTypeToOpen = DirectoryStream. • Else if StreamTypeNameToOpen is "$INDEX_ALLOCATION" then StreamTypeToOpen = DirectoryStream. • Else if CreateOptions.FILE_NON_DIRECTORY_FILE is FALSE, StreamNameToOpen is empty, StreamTypeNameToOpen is empty, Open.File is not NULL, and Open.File.FileType is DirectoryFile then StreamTypeToOpen = DirectoryStream. • Else StreamTypeToOpen = DataStream. • EndIf • EndIf • If StreamTypeToOpen is DirectoryStream: • If StreamTypeNameToOpen is not "$INDEX_ALLOCATION": • If StreamNameToOpen is not empty or StreamTypeNameToOpen is not empty, then the operation MUST be failed with STATUS_NOT_A_DIRECTORY. • EndIf • If Open.File is not NULL and Open.File.FileType is DataFile: • If CreateDisposition == FILE_CREATE then the operation MUST be failed with STATUS_OBJECT_NAME_COLLISION, else the operation MUST be failed with STATUS_NOT_A_DIRECTORY. • EndIf • Else if StreamTypeToOpen is DataStream: • If StreamNameToOpen is empty and Open.File is not NULL and Open.File.FileType is DirectoryFile, the operation MUST be failed with STATUS_FILE_IS_A_DIRECTORY. • EndIf • If PathName contains a trailing backslash: • If StreamTypeToOpen is DataStream or CreateOptions.FILE_NON_DIRECTORY_FILE is TRUE, the operation MUST be failed with STATUS_OBJECT_NAME_INVALID. • EndIf In Section 2.1.5.1.1, Creation of a New File, all instances of FileTypeToOpen were changed to StreamTypeToOpen and all instances of DirectoryFile were changed to DirectoryStream. In that same section, the following was changed from: • File.FileType set to FileTypeToOpen. Changed to:

Errata Published*	Description
	• File.FileType set to DirectoryFile if StreamTypeToOpen is DirectoryStream, else it is set to DataFile. The following was changed from: • If StreamTypeNameToOpen is empty or "$DATA", then the object store MUST create a new data stream for the file as follows: Changed to: • If StreamTypeToOpen is DataStream, then the object store MUST create a new data stream for the file as follows: In Section2.1.5.1.2, Open of an Existing File, all instances of FileTypeToOpen were changed to StreamTypeToOpen, all instances of DirectoryFile were changed to DirectoryStream, all instances of DataFile were changed to DataStream, and all instances of ViewIndexFile were changed to ViewIndexStream.
2018/07/16	In Section 2.1.4.12, Algorithm to Check for an Oplock Break, the following line was changed from: Case OPEN_BREAK_H, as specified in section 2.1.5.1: Changed to: Case OPEN_BREAK_H, as specified in section 2.1.5.1.2:
2018/06/18	In Section 2.1.4.12, Algorithm to Check for an Oplock Break, the following case was added: Case SET_SECURITY, as specified in section 2.1.5.16 Set BreakCacheState to HANDLE_CACHING EndCase In Section 2.1.5.16, Server Requests Setting of Security Information, the following processing rule was added: If Open.Stream.Oplock is not empty, the object store MUST check for an oplock break according to the algorithm in section 2.1.4.12, with input values as follows: Open equal to this operation's Open Oplock equal to Open.Stream.Oplock Operation equal to "SET_SECURITY" OpParams empty

*Date format: YYYY/MM/DD