MS-FSA: Describing Wire Visible Behavior of Microsoft File Systems

Neal Christiansen
Principal Development Lead
Microsoft

What is MS-FSA?

- FSA stands for: File System Algorithms
- An Informative Reference which describes the communication between a server and its underlying object store:
  - Referenced by MS-CIFS, MS-SMB, and MS-SMB2
  - Available on MSDN at: http://msdn.microsoft.com/en-us/library/ff469524(v=PROT.10).aspx

Before MS-FSA

- As the CIFS and SMB protocol documents were developed it became evident that file system specific semantics are exposed through these protocols
  - Originally those file system semantics deemed important enough were documented in the protocol documents themselves.
- With the advent of SMB2 it was realized that continuing to put file system specific semantic into each protocol document was not going to work in the long term.
- File system specific semantics have mostly been removed and replaced with references to MS-FSA across all three protocol documents.

FSBO: File System Behavior Overview

- Developed to address specific questions around file system behavior visible over the wire:
  - Syntax of alternate data streams
  - How Oplocks work
    - Including Win7 Granular Oplocks
  - How ByteRangeLocks work
  - How File Deletion works

FSBO: File System Behavior Overview

- Common error codes returned from top level file system operations
- TimeStamp semantics across all Microsoft file systems
  - Includes what times are maintained
  - Each times resolution
  - What operations update a given timeStamp
- How WildCard evaluation works

FSBO: File System Behavior Overview

- First version available in June 2008
  - Only covered NTFS functionality
- Current version is May 2009
  - Additional functionality documented
  - Added Windows 7 enhancements
  - Updated to cover all Microsoft local File Systems
- Available on MSDN at:
  - http://download.microsoft.com/download/4/3/8/43889780-8d45-4b2e-9d3a-c696a890309f/File%20System%20Behavior%20Overview.pdf

Why MS-FSA?

- Problems with FSBO
  - Not directly applicable to wire visible behavior
    - Written at the level of the Windows file system API
  - Did not provide a comprehensive overview of Windows file systems
    - Only covered specific topics
    - Did not describe why a given error was returned for a given operation
  - Not testable

What is MS-FSA?

- An Informative Reference which describes the communication between a server and its underlying object store
- MS-FSA contains:
  - An Abstract Data Model (ADM) for the underlying object store (3.1.1)
  - A library of common algorithms across operations (3.1.4)
  - An algorithmic description of ~110 wire visible operations (3.1.5)
    - Includes why a given error code is returned
    - Does not define the exact order of the checks
- Developed by multiple members of Microsoft's Storage and File Systems team

What is MS-FSA?

- MS-FSA does not contain:
  - Operations that are not supported remotely
  - The impact local operations may have on remote operations
    - TxF (NTFS support for file level transactions)
    - Memory mapped files
- MS-FSCC: File System Control Codes
  - Important supporting document
  - Defines structures associated with parameters to operations
  - Available on MSDN at: http://msdn.microsoft.com/en-us/library/cc231987(PROT.13).aspx

Current status of MS-FSA

- Publicly available on MSDN
- Preview copy available Nov, 2009
- First release: 03/12/2010
- Current version: 3.0 released 8/27/2010

Examples of how MS-FSA can provide details around difficult to understand operations and concepts in Windows file systems

Security, Access and Share Checks

- Describes the various checks that are made when:
  - Opening an existing file (3.1.5.1.2.1) based on:
    - Type of file (file or directory)
    - Attributes of the file
    - Desired access requested
    - CreateOptions requested
- When creating a new file (3.1.5.1.1)
- When renaming a file (3.1.5.14.11)
- When requesting shared access to a file (3.1.5.1.2.2)

Name Tunneling

- Preserves a set of file characteristics when a file is removed and "quickly" added back (within 15 seconds) to the name space using the same name for both operations:
  - Create processing (3.1.5.1.1)
  - Close processing (3.1.5.4)
  - Rename processing (3.1.5.14.11)
- Characteristics that are tunneled (3.1.1.2):
  - Long file name
  - Short file name
  - Creation time
  - Object ID
- Provides details around what fields are tunneled and when the tunnel cache is checked

Alternate Data Streams

- Allows files and directories to have more than one stream of data
- Describes file creation semantics (3.1.5.1.1):
  - Creating a new file specifying an alternate data stream causes a new empty primary data stream to be automatically created
  - Naming syntax (MS-FSCC section 2.1.5.3)
- Stream rename semantics (3.1.5.14.11.1)
  - Rename of the primary data stream to an alternate data stream causes a new empty primary data stream to be automatically created
  - Can rename a stream over another stream
- Stream deletion semantics (3.1.5.4)

Byte Range Locks

- Supports shared and exclusive locks (3.1.5.7)
- Mandatory (not advisory)
  - Enforced by file system on read (3.1.5.2) and write (3.1.5.3) operations
- If the byte range is not available a lock request can either (3.1.4.10):
  - Wait for range to become available
  - Fail immediately
- Removed via:
  - Explicit unlock (3.1.5.8)
  - File close (3.1.5.4)
- Locks are per stream (3.1.1.5)
- Supports zero-length ranges (3.1.4.10)
- See FSBO for overall byte range lock semantics

USN Journal

- Allows applications to efficiently track which files are being changed:
  - A bitmask of high level reasons is specified (MS-FSCC section 2.3.40)
- Provides details around when USN entries are generated (see references to section 3.1.4.11)

Directory Change Notification

- Allows applications to monitor a directory and optionally its subdirectories for changes (3.1.5.10)
- The following "changes" can be specified as triggers (look for references to 3.1.4.1):
  - File name change
  - Directory name change
  - Attribute change
  - File size change
  - Last write-time change
  - Security change

Oplocks

- Types of oplocks (3.1.1.10 and 3.1.5.17)
  - Level 1 (Read/Write)
  - Level 2 (Read)
  - Batch (Read/Write/Handle)
  - Granular
    - Introduced in Windows 7
- Granted per stream (3.1.1.5)
- How oplocks are broken (3.1.4.12)
- Oplock handshake between object store and server (3.1.5.17.3 and 3.1.5.18)
- See FSBO for overall oplock semantics

Deleting Files

- Files can be deleted in 2 ways:
  - Set DELETE_ON_CLOSE flag when opening the file (3.1.5.1)
  - Set Delete-pending using *FileDispositionInformation* file information class (3.1.5.14.3)
    - Delete-pending state can be toggled on and off
- While in Delete-on-close state the file can still be opened if FILE_SHARE_DELETE access is requested (3.1.5.1)
- While in Delete-pending state attempts to open the file return STATUS_DELETE_PENDING (3.1.5.1)
- The name is not removed from the name space and the contents deleted until the last handle for the given file is closed (3.1.5.4)
- See FSBO for overall deletion semantics

Renaming Files

- Describes details around the following concepts in rename (3.1.5.14.11):
  - Access checks that are performed
  - Handling of case sensitive renames.
  - Handling of superseding renames
  - Handling of cross directory renames

Valid Data Length (VDL)

- File Systems maintain 3 file sizes (3.1.1.5):
  - Allocation size (physical size of file in bytes)
  - File size (logical size of file in bytes)
  - Valid Data Length (portion of file which has been written)
- Allows setting file size without having to write zeroes to newly allocated space
  - A read beyond VDL returns zeroes (3.1.5.2)
- Automatically updated during write (3.1.5.3)
  - Explicitly settable (3.1.5.14.14)

Object ID's

- A GUID that can be associated with a given file
  - File can be opened by ObjectID
    - Not supported remotely
  - Requirements for setting (3.1.5.9.23)
    - Must be unique per volume
  - Requirements for retrieval (3.1.5.9.8)
  - Functionality is optional

Extended Attributes (EA)

- EA's are a name/value pair used to store arbitrary metadata with a file
  - Name is ASCII (not Unicode) (FSCC section 2.4.15)
  - Requirements for setting (3.1.5.14.5)
    - Size of all name/value pairs for a given file is limited to 64K (3.1.5.14.5)
    - Setting a name with no data removes the entry
  - Requirements for retrieval (3.1.5.11.12)
  - Processing at Open time (3.1.5.1.2)
  - Functionality is optional

Reparse Points

- Globally unique 32bit tag plus data which allows an open to be redirected from one path to another
  - Evaluation occurs at open time (3.1.5.1.2)
    - Evaluation can be suppressed
  - Requirements for setting (3.1.5.9.25)
    - Only one reparse point supported per file
    - Size of data limited to 16k
    - Mutually exclusive with EA's
  - Requirements for retrieval (3.1.5.9.9)
  - Requirements for deletion (3.1.5.9.3)
  - Functionality is optional

Symbolic Links and Mount Points

- Implemented using NTFS Reparse Points
  - Requirements for creating (3.1.5.9.25)
    - Symbolic link can have a file type of directory or file
    - Mount Point can only have a file type of directory
    - Symbolic link requires SeCreateSymbolicLinkPrivilege
  - Requirements for retrieval (3.1.5.9.9)
  - Requirements for deletion (3.1.5.9.3)
  - Functionality is optional

Questions?