[MS-SMB]:
Server Message Block (SMB) Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	0.01		MCPP Milestone Longhorn Initial Availability
07/03/2007	1.0	Major	MLonghorn+90
07/20/2007	2.0	Major	Updated and revised the technical content.
08/10/2007	3.0	Major	Updated and revised the technical content.
09/28/2007	4.0	Major	Updated and revised the technical content.
10/23/2007	5.0	Major	Updated and revised the technical content.
11/30/2007	5.0.1	Editorial	Revised and edited the technical content.
01/25/2008	5.0.2	Editorial	Revised and edited the technical content.
03/14/2008	5.0.3	Editorial	Revised and edited the technical content.
05/16/2008	6.0	Major	Updated and revised the technical content.
06/20/2008	7.0	Major	Updated and revised the technical content.
07/25/2008	8.0	Major	Updated and revised the technical content.
08/29/2008	9.0	Major	Updated and revised the technical content.
10/24/2008	10.0	Major	Updated and revised the technical content.
12/05/2008	11.0	Major	Updated and revised the technical content.
01/16/2009	12.0	Major	Updated and revised the technical content.
02/27/2009	13.0	Major	Updated and revised the technical content.
04/10/2009	14.0	Major	Updated and revised the technical content.
05/22/2009	15.0	Major	Updated and revised the technical content.
07/02/2009	16.0	Major	Updated and revised the technical content.
08/14/2009	17.0	Major	Updated and revised the technical content.
09/25/2009	18.0	Major	Updated and revised the technical content.
11/06/2009	19.0	Major	Updated and revised the technical content.
12/18/2009	20.0	Major	Updated and revised the technical content.
01/29/2010	21.0	Major	Updated and revised the technical content.
03/12/2010	22.0	Major	Updated and revised the technical content.
04/23/2010	23.0	Major	Updated and revised the technical content.

Contents

Release: Tuesday, April 6, 2010

1   Introduction

This document defines the Server Message Block (SMB) Version 1.0 Protocol extensions to the Common Internet File System (CIFS) Protocol, which is specified in [MS-CIFS]. The extensions documented here are implemented by Microsoft in Microsoft Windows® versions Microsoft Windows® 2000 operating system, Windows® XP operating system, Windows Server® 2003 operating system, Windows Server® 2003 R2 operating system, Windows Vista® operating system, Windows Server® 2008 operating system, Windows® 7 operating system and Windows Server® 2008 R2 operating system. Unless specifically extended or overridden in this document, all specifications and behaviors that are described for Microsoft Windows NT® operating system clients and servers in [MS-CIFS] apply to the Windows client and server implementations covered in this document. The list of Windows client and server implementations covered in this document is provided in section 6.

Unless otherwise noted, this document only provides the extensions made to the CIFS Protocol relative to the specification in [MS-CIFS]. The extended CIFS Protocol is known as the Server Message Block (SMB) Version 1.0 Protocol. Both this document and [MS-CIFS] are required in order to create a complete implementation of the Server Message Block (SMB) Version 1.0 Protocol.

This document also defines Windows behavior with respect to optional behavior that is described in the specifications of the SMB extensions.

1.1   Glossary

The following terms are defined in [MS-GLOS]:

@GMT Token
8.3 name
Coordinated Universal Time (UTC)
Copychunk Resume Key
discretionary access control list (DACL)
Distributed File System (DFS)
domain
Fid
file allocation table (FAT)
FileId (1)
file system control (FSCTL)
guest account
Information Level
little-endian
message mode
named pipe
network byte order
NT file system (NTFS)
oplock break
opportunistic lock (oplock)
original equipment manufacturer (OEM) character
process identifier (PID)
raw read (on a named pipe)
reparse point
security context (1)
security descriptor
security identifier (SID)
Server Message Block (SMB)
session

Release: Tuesday, April 6, 2010

shadow copy
share
share connect
SMB connection
SMB dialect
SMB session
snapshot
stream
system access control list (SACL)
Transmission Control Protocol (TCP)
Unicode
Unicode string
volume identifier (VolumeId)

The following terms are defined in [MS-CIFS]:

byte mode
Common Internet File System (CIFS)
deprecated
input/output control (IOCTL)
obsolescent
SMB command
SMB message
tree connect

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2   References

1.2.1   Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-CIFS] Microsoft Corporation, "Common Internet File System (CIFS) Protocol Specification", September 2009.

[MS-DFSC] Microsoft Corporation, "Distributed File System (DFS): Referral Protocol Specification", June 2007.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[MS-FSCC] Microsoft Corporation, "File System Control Codes", July 2007.

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions", January 2007.

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification", June 2007.

[MS-RAP] Microsoft Corporation, "Remote Administration Protocol Specification", March 2007.

[MS-SPNG] Microsoft Corporation, "Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO) Protocol Extensions", January 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000, http://www.ietf.org/rfc/rfc2743.txt

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and Ingersoll, W., "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, October 2005, http://www.ietf.org/rfc/rfc4178.txt

1.2.2 Informative References

[MS-BRWS] Microsoft Corporation, "Common Internet File System (CIFS) Browser Protocol Specification", July 2007.

[MS-BRWSA] Microsoft Corporation, "Common Internet File System (CIFS) Browser Auxiliary Protocol Specification", September 2007.

[MS-DFSNM] Microsoft Corporation, "Distributed File System (DFS): Namespace Management Protocol Specification", September 2007.

[MS-FSA] Microsoft Corporation, "File System Algorithms", http://msdn.microsoft.com/en-us/library/860b1516-c452-47b4-bdbc-625d344e2041.aspx

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", January 2007.

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Version 2 Protocol Specification", July 2007.

[MS-WKST] Microsoft Corporation, "Workstation Service Remote Protocol Specification", January 2007.

[MS-WSO] Microsoft Corporation, "Windows System Overview", January 2010.

[MSBRWSE] Thompson IV, D. and McLaughlin, R., "MS Windows NT Browser", https://www.microsoft.com/technet/archive/winntas/deploy/prodspecs/ntbrowse.mspx

[MSDFS] Microsoft Corporation, "How DFS Works", March 2003, http://technet.microsoft.com/en-us/library/cc782417%28WS.10%29.aspx

[MSDN-IMPERS] Microsoft Corporation, "Impersonation" http://msdn.microsoft.com/en-us/library/ms691341.aspx

[NETBEUI] IBM Corporation, "LAN Technical Reference: 802.2 and NetBIOS APIs", 1986, http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/BK8P7001/CCONTENTS

If you have any trouble finding [NETBEUI], please check here.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, http://www.ietf.org/rfc/rfc0793.txt

[RFC1001] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", STD 19, RFC 1001, March 1987, http://www.ietf.org/rfc/rfc1001.txt

[RFC1002] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications", STD 19, RFC 1002, March 1987, http://www.ietf.org/rfc/rfc1002.txt

[SNIA] Storage Networking Industry Association, "Common Internet File System (CIFS) Technical Reference, Revision 1.0", March 2002, http://www.snia.org/tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf

1.3 Overview

Client systems use the Common Internet File System (CIFS) Protocol to request file and print services from server systems over a network. CIFS is a stateful protocol, in which clients establish a **session** with a server and use that session to make a variety of requests to access files, printers, and inter-process communication (IPC) mechanisms, such as **named pipes**. CIFS imposes state to maintain an authentication context, cryptographic operations, file semantics, such as locking, and similar features. A detailed overview of how the CIFS Protocol functions is provided in [MS-CIFS] section 2.

The Server Message Block (SMB) Version 1.0 Protocol extends the CIFS Protocol with additional security, file, and disk management support. These extensions do not alter the basic message sequencing of the CIFS Protocol but introduce new flags, extended requests and responses, and new **Information Levels**. All of these extensions follow a request/response pattern in which the client initiates all of the requests. The base protocol allows for one exception to this pattern--**oplock break**s--as specified in [MS-CIFS] section 3.2.5.42.

This document defines the SMB Version 1.0 Protocol extensions to CIFS, which provide support for the following features:

- New authentication methods, including Kerberos. The Negotiate and Session Setup commands have been enhanced to carry opaque security tokens to support mechanisms that are compatible with the Generic Security Services (GSS).

- Enumeration and access to previous versions of files. A new subcommand that uses a file system control (**FSCTL**) allows the client to query the server for the presence of older versions of files. If the server implements a file system with versioning, then this can be exposed to clients.

- Client requests for server-side data movement operations between files without requiring the data to be read by the client and then written back to the server. As specified in [MS-CIFS], to copy a file on the server requires the client to read all of the data from the server and then write the data back to the server. The SMB Version 1.0 Protocol introduces a method by which such an operation can be done entirely on the server without consuming network resources.

- **SMB connections** that use Direct TCP for the SMB transport. The CIFS Protocol supports the use of NBT for connections, as specified in [MS-CIFS] section 2.1.1.2. The SMB Version 1.0 Protocol includes a method to connect directly over **TCP** (see [RFC793]) without involving NetBIOS (see [RFC1001] and [RFC1002]). Information about NetBIOS is specified in [NETBEUI].

- Support for retrieving extended information in response to **share connect** and file open operations. Certain server functionality and indicators (such as the need for the client to cache the contents of a **share**) are new in the SMB Version 1.0 Protocol and are returned to the client through these extensions to existing commands.

Release: Tuesday, April 6, 2010

- Additional **SMB commands** for the setting and querying of quotas by user. Provided the server supports quotas, the client can constrain the file system capacity consumed by the files of users.

Many of these capabilities are exposed in enhancements to the SMB_COM_NEGOTIATE (section 2.2.4.5) and SMB_COM_SESSION_SETUP_ANDX (section 2.2.4.6) command requests and responses.

1.4 Relationship to Other Protocols

The extensions to the CIFS protocol rely on the Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO), as specified in [RFC4178], for authentication, which in turn relies on Kerberos, as specified in [MS-KILE], and/or the NT LAN Manager (NTLM), as specified in [MS-NLMP], challenge/response authentication protocol.

The Server Message Block (SMB) Version 2 Protocol is a new version of **SMB**. For more information about the SMB Version 2 Protocol, see [MS-SMB2]. This specification does not require implementation of the SMB Version 2 Protocol.

The following protocols extend this specification to provide additional functionality:

- The Distributed File System (DFS): Namespace Referral Protocol, as specified in [MS-DFSC]. For more information, see [MSDFS]. For management of **DFS**, see [MS-DFSNM].

  The following protocols can use the SMB Version 1.0 Protocol as a transport:

  - The Remote Procedure Call (RPC) Protocol Extensions. Note that when named pipes are used, this protocol requires the SMB Protocol. For more information, see [MS-RPCE].

  - The Remote Mailslot Protocol. This protocol can use the SMB Version 1.0 Protocol as a transport but supports other transports as well. For more information, see [MS-MAIL].

  - The CIFS Browser Protocol. This protocol uses the Remote Mailslot Protocol and the RAP as transport protocols, which in turn can use this specification. It does not use this specification directly, but is included here for completeness. For more information, see [MS-BRWS] and [MSBRWSE].

The following diagram illustrates the relationship amongst the protocols.
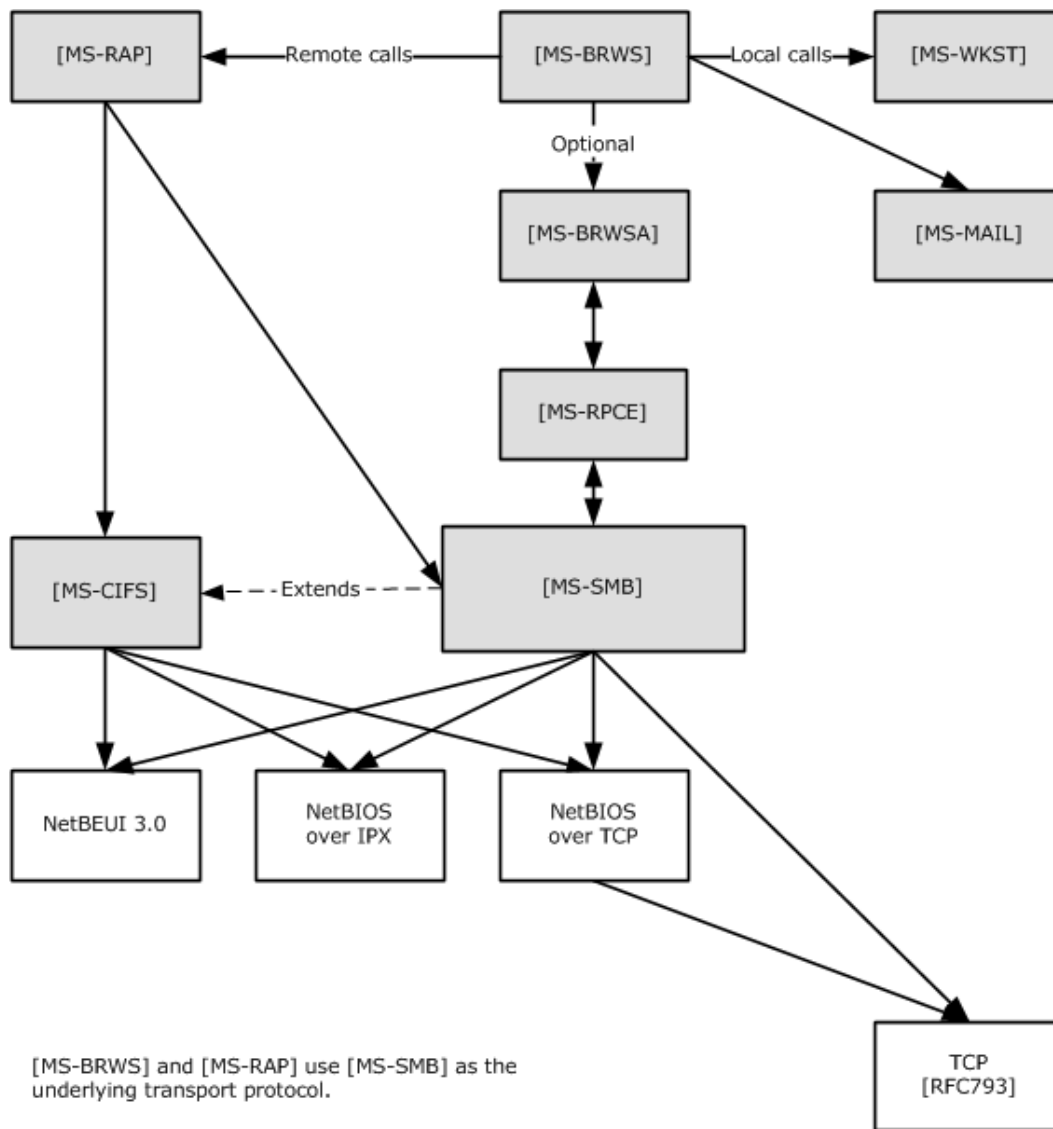
Figure 1: Relationships to other protocols

1.5 Prerequisites/Preconditions

The SMB Version 1.0 Protocol assumes the availability of the following resources:

- An underlying transport protocol that supports reliable, in-order message delivery.

- An underlying object store on the server, such as a file system, exposing file, named pipe, or printer objects.

1.6 Applicability Statement

The extensions specified in this document are applicable to environments in which the security characteristics of the base protocol, as specified in [MS-CIFS], are insufficient. In particular, these extensions provide for enhanced message integrity and stronger authentication mechanisms.

The extensions are applicable to an environment that requires tighter data retention policies. In particular, through the use of previous version capabilities, the extensions allow access to versions of a file that have been changed or deleted when the server supports this capability. This feature is applicable to environments that require more stringent data retention policies that include maintaining access to previous versions of files.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- Supported Transports: The extensions in this document add additional transports, as defined in section 2.1.

- Security and Authentication Methods: The extensions in this document add additional authentication methods, as specified in section 3.2.4.2.

- Capability Negotiation: The extensions in this document use capability negotiation, as specified in [MS-CIFS] sections 1.7, 2.2.4.52, 2.2.4.53, and 3.3.1.2.

SMB dialect negotiation is handled as specified in [MS-CIFS] sections 1.7 and 3.2.4.2. The extensions specified in this document introduce no new dialects and apply only to connections that have negotiated the **NT LAN Manager** dialect, as identified by the "NT LM 0.12" dialect identification string. The extensions specified in this document are detected via the following methods:

- They can be returned in the **Capabilities** field, as specified in [MS-CIFS] section 2.2.4.52. Specific new capability options are defined in this document.

- They can be supplied or returned in the **Flags** and **Flags2** fields of the SMB header, as specified in [MS-CIFS] sections 2.2.3.1.

- A server can return an error code (STATUS_NOT_SUPPORTED) when a client request is sent to a server for a new feature that is not supported.

A client written to support these extensions cannot require that the target server implement these extensions to successfully connect. Thus, a server that does not implement an extension is still accessible by a client that implements that extension, although the relevant new features might not be available. The one exception is that a client offers the capability to be configured to require the new security features to create a more secure environment so that the client could be restricted from connecting successfully to servers that do not implement these features.

Negotiation of the use of the Generic Security Service Application Program Interface (GSS API) for authentication is specified in section 3.2.4.2.4. The GSS API is specified in [RFC2743].

1.8 Vendor-Extensible Fields

The CAP_UNIX capability bit is specified in order to allow third-party implementers to collaborate on the definition of a specific set of extensions. SMB_COM_TRANSACTION2 Information Levels in the range 0x200 to 0x3E0 (inclusive) are reserved for these extensions.<1>

1.9 Standards Assignments

In addition to any standards assignments specified in [MS-CIFS], the Direct TCP Transport, as specified in section 2.2, makes use of the following assignment:

Parameter	TCP port value	Reference
Microsoft-DS	445 (0x01BD)	[IANAPORT]

SMB transports can have assigned port numbers or other assigned values. See the documentation for the specific transport for more information.

2   Messages

An SMB Version 1.0 Protocol implementation MUST implement CIFS, as specified by section 2 of the [MS-CIFS] specification.

2.1   Transport

In addition to the transport protocols listed in section 2.1 of [MS-CIFS], the extended version of the protocol supports the use of TCP as a transport layer. Hereafter, the special TCP-related characteristics that are employed in the application of SMB over TCP are known as the Direct TCP transport.<2>

The extended version of the SMB Version 1.0 Protocol can use Direct TCP over either IPv4 or IPv6 as a reliable stream-oriented transport for **SMB messages**. No NetBIOS layer is provided or used. TCP provides a full, duplex, sequenced, and reliable transport for the connection. When using TCP as the reliable connection-oriented transport, the extended version of the SMB Version 1.0 Protocol makes no higher-level attempts to ensure sequenced delivery of messages between a client and server. The TCP transport has mechanisms to detect failures of either the client node or the server node, and to deliver such an indication to the client or server software so that it can clean up the state.

When using Direct TCP as the SMB transport, the implementer MUST establish a TCP connection from an SMB client to a TCP port on the server. The TCP source port used by the SMB client MAY be any TCP port value. The SMB server SHOULD listen for connections on port 445. This port number has been registered with the Internet Assigned Numbers Authority (IANA) and has been officially assigned for Microsoft-DS.<3>

When using Direct TCP as the SMB transport, the implementer MUST prepend a 4-byte Direct TCP transport packet header to each SMB message. This transport header MUST be formatted as a byte of zero (8 zero bits) followed by 3 bytes that indicate the length of the SMB message that is encapsulated. The body of the SMB packet follows as a variable-length payload. A Direct TCP transport packet has the following structure (in **network byte order**).

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Zero								Stream Protocol Length																							
SMB Message (variable)																															
...																															

Zero (1 byte): The first byte of the Direct TCP transport packet header MUST be zero (0x00).

Stream Protocol Length (3 bytes): The length, in bytes, of the SMB message. This length is formatted as a 3-byte integer in network byte order. The length field does not include the 4-byte Direct TCP transport header; rather, it is only the length of the enclosed SMB message.<4>

SMB Message (variable): The body of the SMB packet. The length of an SMB message varies based on the SMB command represented by the message.

2.2   Message Syntax

A client exchanges messages with a server to access resources on the server. These messages are called SMB messages or SMBs. Every SMB message has a common format, as defined in [MS-CIFS] section 2.2.

All SMB messages MUST begin with a fixed-length SMB header (as specified in [MS-CIFS], section 2.2.1). The header contains a command field that indicates the operation code that the client requests or to which the server responds. An SMB message is of variable length. The actual length depends on the SMB command field (and consequent appended data structures) and whether the SMB message is a client request or a server response.

Unless otherwise indicated, numeric fields are integers of the specified byte length.

Unless otherwise specified, multibyte fields (that is, 16-bit, 32-bit, and 64-bit fields) in an SMB message MUST be transmitted in **little-endian** byte order (least significant byte first).

Unless otherwise noted, fields marked as Reserved SHOULD be set to zero when being sent and MUST be ignored upon receipt. Unless otherwise noted, unused or reserved bits in bit fields SHOULD be set to zero when being sent and MUST be ignored upon receipt.

When an error occurs, unless otherwise noted in this specification, an SMB server SHOULD return an Error Response message. An Error Response message is comprised of a complete SMB header, along with an empty parameter and data portion.<5>

2.2.1   Common Data Type Extensions

2.2.1.1   Character Sequences

2.2.1.1.1   Pathname Extensions

In addition to the specification in [MS-CIFS] section 2.2.1.1.2, pathnames include the following extension:

- Previous Version Tokens -- Support for this feature is optional.

Pathnames are allowed to contain a previous version token (or **@GMT token**), as a directory element in a path. A previous version token indicates that the pathname is a request to access the previous version (or **shadow copy**) of the file or directory at a particular point in time. This feature is available on any path-based operation (for example, SMB_COM_NT_CREATE_ANDX). A pathname MUST NOT contain more than one previous version token.

For example, requesting a previous version of the file \\server\mydocs\reviews\feb01.doc at 2:44:00 P.M. on March30, 2001 **UTC** is specified in the following format:

```
\\server\mydocs\reviews\@GMT-2001.03.30-14.44.00\feb01.doc
```

The same technique can be used to build a path that represents a previous version of a directory as opposed to a file.

For example, requesting a previous version of the directory \\server\mydocs\reviews at 2:44:00 PM on 3/30/01 UTC can be specified in either of the following formats:

A token appearing as an intermediate path component:

```
\\server\mydocs\@GMT-2001.03.30-14.44.00\reviews
```

A token appearing as a final path component:

```
\\server\mydocs\reviews\@GMT-2001.03.30-14.44.00
```

In addition, it is possible to request an enumeration of available previous version timestamps (or **snapshots**) of a file or directory. While the NT_TRANSACT_IOCTL subcommand can be used with the FSCTL_SRV_ENUMERATE_SNAPSHOTS FSCTL code to enumerate available previous version timestamps using a valid **Fid** (section 2.2.7.2.1), these extensions also present a path-based method to access this functionality. The TRANS2_FIND_FIRST2 subcommand's SMB_FIND_FILE_BOTH_DIRECTORY_INFO Information Level (section 2.2.6.1) has been extended to allow a special previous version wildcard token, @GMT-*.

For example, requesting an enumeration of available previous version timestamps of the examples, discussed earlier in this section, can be specified in the following ways:

```
\\server\mydocs\reviews\@GMT-*\feb01.doc
\\server\mydocs\@GMT-*\reviews
\\server\mydocs\reviews\@GMT-*
```

2.2.1.2   File Attributes

2.2.1.2.1   Extended File Attribute (SMB_EXT_FILE_ATTR) Extensions

The list of extended file attributes valid in 32-bit attribute values, as specified in [MS-CIFS] section 2.2.1.2.3, has been extended to include the following attributes:

- ATTR_SPARSE

- ATTR_REPARSE_POINT

- ATTR_OFFLINE

- ATTR_NOT_CONTENT_INDEXED

- ATTR_ENCRYPTED

The following table lists all possible values. Unless otherwise noted, any combination of these values is acceptable.

Name & bitmask	Extension	Meaning
ATTR_READONLY 0x00000001	No	File is read-only. Applications cannot write or delete the file.
ATTR_HIDDEN 0x00000002	No	File is hidden. It is not to be included in an ordinary directory enumeration.
ATTR_SYSTEM 0x00000004	No	File is part of or is used exclusively by the operating system.

Name & bitmask	Extension	Meaning
ATTR_DIRECTORY 0x00000010	No	File is a directory.
ATTR_ARCHIVE 0x00000020	No	File has not been archived since it was last modified.
ATTR_NORMAL 0x00000080	No	File has no other attributes set. This value is valid only when used alone.
ATTR_TEMPORARY 0x00000100	No	File is temporary.
ATTR_SPARSE 0x00000200	Yes	File is a sparse file.
ATTR_REPARSE_POINT 0x00000400	Yes	File or directory has an associated **reparse point**.
ATTR_COMPRESSED 0x00000800	No	File is compressed on the disk. This does not affect how it is transferred over the network.
ATTR_OFFLINE 0x00001000	Yes	File data is not available. The attribute indicates that the file has been moved to offline storage.
ATTR_NOT_CONTENT_INDEXED 0x00002000	Yes	File or directory SHOULD NOT be indexed by a content indexing service.
ATTR_ENCRYPTED 0x00004000	Yes	File or directory is encrypted. For a file, this means that all data in the file is encrypted. For a directory, this means that encryption is the default for newly created files and subdirectories.
Reserved 0xFFFF8048	N/A	SHOULD be set to zero when sending and MUST be ignored upon receipt of the message.

2.2.1.2.2   File System Attribute Extensions

The list of file system attributes, as specified in [MS-CIFS] section 2.2.8.2.6, has been extended. For completeness, the following table lists all of the available attribute flags and their symbolic constants. Unless otherwise noted, any combination of the following bits is valid. Any bit that is not listed in this section is considered reserved; the sender SHOULD set it to zero, and the receiver MUST ignore it. For more information, see [MS-FSCC] section 2.5.1.

Name & bitmask	Extension	Meaning
FILE_CASE_SENSITIVE_SEARCH 0x00000001	No	File system supports case-sensitive file names.
FILE_CASE_PRESERVED_NAMES 0x00000002	No	File system preserves the case of file names when it stores the name on disk.
FILE_UNICODE_ON_DISK	No	File system supports **Unicode** in file names.

Name & bitmask	Extension	Meaning
0x00000004		
FILE_PERSISTENT_ACLS 0x00000008	No	File system preserves and enforces access control lists.
FILE_FILE_COMPRESSION 0x00000010	No	File system supports file-based compression. This flag is incompatible with FILE_VOLUME_IS_COMPRESSED. This flag does not affect how data is transferred over the network.
FILE_VOLUME_QUOTAS 0x00000020	Yes	File system supports per-user quotas.
FILE_SUPPORTS_SPARSE_FILES 0x00000040	Yes	File system supports sparse files.
FILE_SUPPORTS_REPARSE_POINTS 0x00000080	Yes	File system supports reparse points.
FILE_SUPPORTS_REMOTE_STORAGE 0x00000100	Yes	File system supports remote storage.
FILE_VOLUME_IS_COMPRESSED 0x00008000	No	Volume is a compressed volume. This flag is incompatible with FILE_FILE_COMPRESSION. This does not affect how data is transferred over the network.
FILE_SUPPORTS_OBJECT_IDS 0x00010000	Yes	File system supports object identifiers.
FILE_SUPPORTS_ENCRYPTION 0x00020000	Yes	File system supports encryption.
FILE_NAMED_STREAMS 0x00040000	Yes	File system supports multiple named data **streams** for a file.
FILE_READ_ONLY_VOLUME 0x00080000	Yes<6>	Specified volume is read-only.
FILE_SEQUENTIAL_WRITE_ONCE 0x00100000	Yes<7>	Specified volume can be written to one time only. The write MUST be performed in sequential order.
FILE_SUPPORTS_TRANSACTIONS 0x00200000	Yes<8>	File system supports transaction processing.
FILE_SUPPORTS_HARD_LINKS 0x00400000	Yes<9>	File system supports direct links to other devices and partitions.
FILE_SUPPORTS_EXTENDED_ATTRIBUTES 0x00800000	Yes<10>	File system supports extended attributes (EAs).
FILE_SUPPORTS_OPEN_BY_FILE_ID	Yes<11>	File system supports open by **FileID**.

Name & bitmask	Extension	Meaning
0x01000000		
FILE_SUPPORTS_USN_JOURNAL 0x02000000	Yes<12>	File system supports update sequence number (USN) journals.
Reserved 0xFE007E00	N/A	These bits fields SHOULD be set to zero when sending and MUST be ignored when the message is received.

2.2.1.3 Unique Identifiers

The SMB Version 1.0 Protocol makes use of the following data types from [MS-DTYP]:

▪ GUID as specified in section 2.3.2.1

The list of unique identifiers, specified in [MS-CIFS] section 2.2.1.6, has been extended to include the following new unique identifiers:

▪ 64-bit file identifier (FileId)

▪ Volume GUID (VolumeGUID)

▪ **Copychunk Resume Key**

2.2.1.3.1 FileId Generation

64-bit file identifiers (FileIds) are generated on SMB servers. The generation of FileIds MUST satisfy the following constraints:

▪ The FileId MUST be a 64-bit opaque value.

▪ The FileId MUST be unique for a file on a given object store.<13>

▪ The FileId for a file MUST persist for the lifetime of a file on a given object store. A FileId MUST NOT be changed when a file is renamed. When the file is deleted, the FileId MAY be reused.

▪ All possible values for FileId are valid.

2.2.1.3.2 VolumeGUID Generation

VolumeGUIDs (Volume Globally Unique Identifiers, or **volume identifiers**, see also [MS-DTYP] section 2.3.2) are generated on SMB servers. The generation of VolumeGUIDs MUST satisfy the following constraints:

▪ The VolumeGUID MUST be a 128-bit opaque value.

▪ The VolumeGUID MUST be unique for a logical file system volume on a given server.

▪ The VolumeGUID for the volume can change while the system is running. The VolumeGUID can change when the system is restarted.

▪ All possible values for the VolumeGUID are valid.

2.2.1.3.3   Copychunk Resume Key Generation

Copychunk Resume Keys are generated on SMB servers. The generation of Copychunk Resume Keys MUST satisfy the following constraints:

- The Copychunk Resume Key MUST be a 24-byte opaque value generated by an SMB server in response to a request by the client (an SMB_COM_NT_TRANSACTION request with an NT_TRANSACT_IOCTL subcommand for the FSCTL_SRV_REQUEST_RESUME_KEY). For more information, see section 2.2.7.2.

- The Copychunk Resume Key MUST be unique on the SMB server for a given open file on a server.

- The Copychunk Resume Key MUST remain valid for the lifetime of the open file on the server.

- All possible values for the Copychunk Resume Key are valid.

2.2.1.4   Access Masks

The SMB protocol introduces the use of Access Mask structures, which are based on the ACCESS_MASK data type specified in [MS-DTYP] section 2.4.3. SMB defines two types of access masks for two basic groups: either for a file, pipe, or printer (specified in section 2.2.1.4.1) or for a directory (specified in section 2.2.1.4.2). Each access mask MUST be a combination of zero or more of the bit positions.

2.2.1.4.1   File_Pipe_Printer_Access_Mask

The following SMB Access Mask structure is defined for use on a file, named pipe, or printer.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
File_Pipe_Printer_Access_Mask																															

File_Pipe_Printer_Access_Mask (4 bytes): For a file, named pipe, or printer, the value MUST be constructed using the following values (for a printer, the value MUST have at least one of the following: FILE_WRITE_DATA, FILE_APPEND_DATA, or GENERIC_WRITE).

Value	Meaning
FILE_READ_DATA 0x00000001	This value indicates the right to read data from the file or named pipe.
FILE_WRITE_DATA 0x00000002	This value indicates the right to write data into the file or named pipe beyond the end of the file.
FILE_APPEND_DATA 0x00000004	This value indicates the right to append data into the file or named pipe.
FILE_READ_EA 0x00000008	This value indicates the right to read the extended attributes of the file or named pipe.
FILE_WRITE_EA 0x00000010	This value indicates the right to write or change the extended attributes to the file or named pipe.

Value	Meaning
FILE_EXECUTE 0x00000020	This value indicates the right to execute the file.
FILE_READ_ATTRIBUTES 0x00000080	This value indicates the right to read the attributes of the file.
FILE_WRITE_ATTRIBUTES 0x00000100	This value indicates the right to change the attributes of the file.
DELETE 0x00010000	This value indicates the right to delete the file.
READ_CONTROL 0x00020000	This value indicates the right to read the security descriptor for the file or named pipe.
WRITE_DAC 0x00040000	This value indicates the right to change the **discretionary access control list (DACL)** in the **security descriptor** for the file or named pipe. For the DACL data structure, see ACL in [MS-DTYP] section 2.4.5.
WRITE_OWNER 0x00080000	This value indicates the right to change the owner in the security descriptor for the file or named pipe.
SYNCHRONIZE 0x00100000	This flag SHOULD be ignored by both clients and servers.
ACCESS_SYSTEM_SECURITY 0x01000000	This value indicates the right to read or change the **system access control list (SACL)** in the security descriptor for the file or named pipe. For the SACL data structure, see ACL in [MS-DTYP] section 2.4.5.
MAXIMUM_ALLOWED 0x02000000	This value indicates that the client is requesting an **open** to the file with the highest level of access the client has on this file. If no access is granted for the client on this file, the server MUST fail the open with STATUS_ACCESS_DENIED.
GENERIC_ALL 0x10000000	This value indicates a request for all the access flags that are previously listed, except MAXIMUM_ALLOWED and ACCESS_SYSTEM_SECURITY.
GENERIC_EXECUTE 0x20000000	This value indicates a request for the following combination of access flags listed above: FILE_READ_ATTRIBUTES, FILE_EXECUTE, SYNCHRONIZE, and READ_CONTROL.
GENERIC_WRITE 0x40000000	This value indicates a request for the following combination of access flags listed above: FILE_WRITE_DATA, FILE_APPEND_DATA, FILE_WRITE_ATTRIBUTES, FILE_WRITE_EA, SYNCHRONIZE, and READ_CONTROL.
GENERIC_READ 0x80000000	This value indicates a request for the following combination of access flags listed above: FILE_READ_DATA, FILE_READ_ATTRIBUTES, FILE_READ_EA, SYNCHRONIZE, and READ_CONTROL.

2.2.1.4.2   Directory_Access_Mask

The following SMB Access Mask is defined for use on a directory.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
															Directory_Access_Mask																

Directory_Access_Mask (4 bytes): For a directory, the value MUST be constructed using the following values:

Value	Meaning
FILE_LIST_DIRECTORY 0x00000001	This value indicates the right to enumerate the contents of the directory.
FILE_ADD_FILE 0x00000002	This value indicates the right to create a file under the directory.
FILE_ADD_SUBDIRECTORY 0x00000004	This value indicates the right to add a sub-directory under the directory.
FILE_READ_EA 0x00000008	This value indicates the right to read the extended attributes of the directory.
FILE_WRITE_EA 0x00000010	This value indicates the right to write or change the extended attributes of the directory.
FILE_TRAVERSE 0x00000020	This value indicates the right to traverse this directory if the server enforces traversal checking.
FILE_DELETE_CHILD 0x00000040	This value indicates the right to delete the files and directories within this directory.
FILE_READ_ATTRIBUTES 0x00000080	This value indicates the right to read the attributes of the directory.
FILE_WRITE_ATTRIBUTES 0x00000100	This value indicates the right to change the attributes of the directory.
DELETE 0x00010000	This value indicates the right to delete the directory.
READ_CONTROL 0x00020000	This value indicates the right to read the security descriptor for the directory.
WRITE_DAC 0x00040000	This value indicates the right to change the DACL in the security descriptor for the directory. For the DACL data structure, see ACL in [MS-DTYP] section 2.4.5.
WRITE_OWNER 0x00080000	This value indicates the right to change the owner in the security descriptor for the directory.
SYNCHRONIZE 0x00100000	This flag SHOULD be ignored by both clients and servers.

Value	Meaning
ACCESS_SYSTEM_SECURITY 0x01000000	This value indicates the right to read or change the SACL in the security descriptor for the directory. For the SACL data structure, see ACL in [MS-DTYP] section 2.4.5.
MAXIMUM_ALLOWED 0x02000000	This value indicates that the client is requesting an open to the directory with the highest level of access that the client has on this directory. If no access is granted for the client on this directory, then the server MUST fail the open with STATUS_ACCESS_DENIED.
GENERIC_ALL 0x10000000	This value indicates a request for all of the access flags that are listed above, except MAXIMUM_ALLOWED and ACCESS_SYSTEM_SECURITY.
GENERIC_EXECUTE 0x20000000	This value indicates a request for the following access flags listed above: FILE_READ_ATTRIBUTES, FILE_TRAVERSE, SYNCHRONIZE, and READ_CONTROL.
GENERIC_WRITE 0x40000000	This value indicates a request for the following access flags listed above: FILE_ADD_FILE, FILE_ADD_SUBDIRECTORY, FILE_WRITE_ATTRIBUTES, FILE_WRITE_EA, SYNCHRONIZE, and READ_CONTROL.
GENERIC_READ 0x80000000	This value indicates a request for the following access flags listed above: FILE_LIST_DIRECTORY, FILE_READ_ATTRIBUTES, FILE_READ_EA, SYNCHRONIZE, and READ_CONTROL.

2.2.2   Defined Constant Extensions

2.2.2.1   SMB_COM Command Codes

No new defined constants are introduced for this section.<14>

2.2.2.2   Transaction Subcommand Codes

In addition to the transaction subcommand codes specified in [MS-CIFS] section 2.2.2.2, the following modifications and extensions apply. In the following tables, the Description column is also used to specify changes in a particular subcommand's current usage status.

Transaction Codes used with SMB_COM_TRANSACTION

Constant/value	Description
TRANS_RAW_READ_NMPIPE 0x0036	This command code has changed from **deprecated** to **obsolescent**.
TRANS_CALL_NMPIPE 0x0054	This command code has changed from current to obsolescent.

Transaction Codes used with SMB_COM_TRANSACTION2

Constant/value	Description
TRANS2_SET_FS_INFORMATION	Set information on a file system on the server. This command code has

Constant/value	Description
0x0004	changed from reserved but not implemented to current.

Transaction Codes used with SMB_COM_NT_TRANSACT

Constant/value	Description
NT_TRANSACT_QUERY_QUOTA 0x0007	Query a server for a user's disk quota information. This command code is new to these extensions.
NT_TRANSACT_SET_QUOTA 0x0008	Set a user's disk quota information on a server. This command code is new to these extensions.
NT_TRANSACT_CREATE2 0x0009	This command code is new to these extensions. The server processes and responds to the NT_TRANSACT_CREATE2 command the same way it would for an NT_TRANSACT_CREATE command, as specified in [MS-CIFS] section 3.3.5.58.1.

2.2.2.3   Information Level Codes

The following new Information Level codes are specified in addition to those defined in [MS-CIFS] section 2.2.2.3.

2.2.2.3.1   FIND Information Level Codes

The following new Information Level codes are specified in addition to those specified in [MS-CIFS] section 2.2.2.3.1.<15>

Name	Code	Meaning	Dialect
SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO	0x0105	Returns the SMB_FIND_FULL_DIRECTORY_INFO data with a FileId.	NT LANMAN
SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO	0x0106	Returns the SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO data with a FileId.	NT LANMAN

2.2.2.3.2   QUERY_FS Information Level Codes

No new SMB-specific Information Level codes are specified for these extensions.

2.2.2.3.3   QUERY Information Level Codes

No new SMB-specific Information Level codes are specified for these extensions.

2.2.2.3.4   SET Information Level Codes

No new SMB-specific Information Level codes are specified for these extensions.

2.2.2.3.5 Pass-through Information Level Codes

This document provides an extension of a new Information Level code value range called **pass-through Information Levels**, which can be used to set or query information on the server. These Information Levels allow SMB clients to directly query Information Levels native to the underlying object store.<16>

Servers indicate support for these new pass-through Information Levels by setting the new CAP_INFOLEVEL_PASSTHRU capability flag in an SMB_COM_NEGOTIATE server response (section 2.2.4.5.2).

To access these new Information Levels, a client adds the constant SMB_INFO_PASSTHROUGH (0x03e8) to the desired native information class level value. This value is then sent in the **InformationLevel** field of the particular SMB_COM_TRANSACTION2 subcommand being used to access the Information Levels.

2.2.2.3.6 Other Information Level Codes

In addition, SMB_COM_TRANSACTION2 Information Levels in the range 0x200 to 0x3E0 (inclusive) are reserved for third-party extensions, as described in section 1.8.<17>

2.2.2.4 SMB Error Classes and Codes

The following is a list of 32-bit status codes that are required to implement these extensions, their associated values, and a description of what they represent.<18>

NT status value	Description
0x00000000 STATUS_SUCCESS	The client request is successful.
0x00010002 STATUS_INVALID_SMB	An invalid SMB client request is received by the server.
0x00050002 STATUS_SMB_BAD_TID	The client request received by the server contains an invalid TID value.
0x00160002 STATUS_SMB_BAD_COMMAND	The client request received by the server contains an unknown SMB command code.
0x005B0002 STATUS_SMB_BAD_UID	The client request to the server contains an invalid UID value.
0x00FB0002 STATUS_SMB_USE_STANDARD	The client request received by the server is for a non-standard SMB operation (for example, an SMB_COM_READ_MPX request on a non-disk share). The client SHOULD send another request with a different SMB command to perform this operation.
0x80000005 STATUS_BUFFER_OVERFLOW	The data was too large to fit into the specified buffer.
0x80000006 STATUS_NO_MORE_FILES	No more files were found that match the file specification.
0x8000002D STATUS_STOPPED_ON_SYMLINK	The create operation stopped after reaching a symbolic link.

NT status value	Description
0xC0000002 STATUS_NOT_IMPLEMENTED	The requested operation is not implemented.
0xC000000D STATUS_INVALID_PARAMETER	The parameter specified in the request is not valid.
0xC000000E STATUS_NO_SUCH_DEVICE	A device that does not exist was specified.
0xC0000010 STATUS_INVALID_DEVICE_REQUEST	The specified request is not a valid operation for the target device.
0xC0000016 STATUS_MORE_PROCESSING_REQUIRED	If extended security has been negotiated, then this error code can be returned in the SMB_COM_SESSION_SETUP_ANDX response from the server to indicate that additional authentication information is to be exchanged. See section 2.2.4.6 for details.
0xC0000022 STATUS_ACCESS_DENIED	The client did not have the required permission needed for the operation.
0xC0000023 STATUS_BUFFER_TOO_SMALL	The buffer is too small to contain the entry. No information has been written to the buffer.
0xC0000034 STATUS_OBJECT_NAME_NOT_FOUND	The object name is not found.
0xC0000035 STATUS_OBJECT_NAME_COLLISION	The object name already exists.
0xC000003A STATUS_OBJECT_PATH_NOT_FOUND	The path to the directory specified was not found. This error is also returned on a create request if the operation requires the creation of more than one new directory level for the path specified.
0xC00000A5 STATUS_BAD_IMPERSONATION_LEVEL	A specified impersonation level is invalid. This error is also used to indicate that a required impersonation level was not provided.
0xC00000B5 STATUS_IO_TIMEOUT	The specified I/O operation was not completed before the time-out period expired.
0xC00000BA STATUS_FILE_IS_A_DIRECTORY	The file that was specified as a target is a directory and the caller specified that it could be anything but a directory.
0xC00000BB STATUS_NOT_SUPPORTED	The client request is not supported.
0xC00000C9 STATUS_NETWORK_NAME_DELETED	The network name specified by the client has been deleted on the server. This error is returned if the client specifies an incorrect TID or the share on the server represented by the TID was deleted.
0xC0000203 STATUS_USER_SESSION_DELETED	The user session specified by the client has been deleted on the server. This error is returned by the server if the client sends an incorrect UID.

NT status value	Description
0xC000035C STATUS_NETWORK_SESSION_EXPIRED	The client's session has expired; therefore, the client MUST re-authenticate to continue accessing remote resources.
0xC000205A STATUS_SMB_TOO_MANY_UIDS	The client has requested too many UID values from the server or the client already has an **SMB session** setup with this UID value.

2.2.2.5 Session Key Protection Hash

The SSKeyHash is a well-known constant array.

```
BYTE SSKeyHash[256] = {
0x53, 0x65, 0x63, 0x75, 0x72, 0x69, 0x74, 0x79,
0x20, 0x53, 0x69, 0x67, 0x6e, 0x61, 0x74, 0x75,
0x72, 0x65, 0x20, 0x4b, 0x65, 0x79, 0x20, 0x55,
0x70, 0x67, 0x72, 0x61, 0x64, 0x65, 0x79, 0x07,
0x6e, 0x28, 0x2e, 0x69, 0x88, 0x10, 0xb3, 0xdb,
0x01, 0x55, 0x72, 0xfb, 0x74, 0x14, 0xfb, 0xc4,
0xc5, 0xaf, 0x3b, 0x41, 0x65, 0x32, 0x17, 0xba,
0xa3, 0x29, 0x08, 0xc1, 0xde, 0x16, 0x61, 0x7e,
0x66, 0x98, 0xa4, 0x0b, 0xfe, 0x06, 0x83, 0x53,
0x4d, 0x05, 0xdf, 0x6d, 0xa7, 0x51, 0x10, 0x73,
0xc5, 0x50, 0xdc, 0x5e, 0xf8, 0x21, 0x46, 0xaa,
0x96, 0x14, 0x33, 0xd7, 0x52, 0xeb, 0xaf, 0x1f,
0xbf, 0x36, 0x6c, 0xfc, 0xb7, 0x1d, 0x21, 0x19,
0x81, 0xd0, 0x6b, 0xfa, 0x77, 0xad, 0xbe, 0x18,
0x78, 0xcf, 0x10, 0xbd, 0xd8, 0x78, 0xf7, 0xd3,
0xc6, 0xdf, 0x43, 0x32, 0x19, 0xd3, 0x9b, 0xa8,
0x4d, 0x9e, 0xaa, 0x41, 0xaf, 0xcb, 0xc6, 0xb9,
0x34, 0xe7, 0x48, 0x25, 0xd4, 0x88, 0xc4, 0x51,
0x60, 0x38, 0xd9, 0x62, 0xe8, 0x8d, 0x5b, 0x83,
0x92, 0x7f, 0xb5, 0x0e, 0x1c, 0x2d, 0x06, 0x91,
0xc3, 0x75, 0xb3, 0xcc, 0xf8, 0xf7, 0x92, 0x91,
0x0b, 0x3d, 0xa1, 0x10, 0x5b, 0xd5, 0x0f, 0xa8,
0x3f, 0x5d, 0x13, 0x83, 0x0a, 0x6b, 0x72, 0x93,
0x14, 0x59, 0xd5, 0xab, 0xde, 0x26, 0x15, 0x6d,
0x60, 0x67, 0x71, 0x06, 0x6e, 0x3d, 0x0d, 0xa7,
0xcb, 0x70, 0xe9, 0x08, 0x5c, 0x99, 0xfa, 0x0a,
0x5f, 0x3d, 0x44, 0xa3, 0x8b, 0xc0, 0x8d, 0xda,
0xe2, 0x68, 0xd0, 0x0d, 0xcd, 0x7f, 0x3d, 0xf8,
0x73, 0x7e, 0x35, 0x7f, 0x07, 0x02, 0x0a, 0xb5,
0xe9, 0xb7, 0x87, 0xfb, 0xa1, 0xbf, 0xcb, 0x32,
0x31, 0x66, 0x09, 0x48, 0x88, 0xcc, 0x18, 0xa3,
0xb2, 0x1f, 0x1f, 0x1b, 0x90, 0x4e, 0xd7, 0xe1
};
```

2.2.3 SMB Message Structure Extensions

2.2.3.1 SMB Header Extensions

All client requests MUST begin with a fixed-size SMB header, as specified in [MS-CIFS] section 2.2.3.1. All server responses, with the exception of the SMB_COM_READ_RAW response message, as specified in [MS-CIFS] section 2.2.4.22.2, MUST begin with the same fixed-size SMB header.

```
SMB_Header
  {
  UCHAR  Protocol[4];
  UCHAR  Command;
  SMB_ERROR Status;
  UCHAR  Flags;
  USHORT Flags2;
  USHORT PIDHigh;
  UCHAR  SecurityFeatures[8];
  USHORT Reserved;
  USHORT TID;
  USHORT PIDLow;
  USHORT UID;
  USHORT MID;
  }
```

The following SMB header fields contain extensions:

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Protocol																															
Command								Status																							
...								Flags								Flags2																
PIDHigh																SecurityFeatures															
...																															
...																Reserved															
TID																PIDLow															
UID																MID															

Flags2 (2 bytes): The **Flags2** field contains individual bit flags that, depending on the negotiated SMB dialect, indicate various client and server capabilities. This field is defined as specified in [MS-CIFS] section 2.2.3.1. There are several new Flags2 values in the SMB header that are not in [MS-CIFS], but are part of these extensions. Unused bit fields SHOULD<19> be set to zero by the sender when sending an SMB message and MUST be ignored when received by the receiver. This field is constructed using the values listed in section 2.2.3.1 of [MS-CIFS], as well as the following additional values:

Name & bitmask	Meaning
SMB_FLAGS2_COMPRESSED	If set by the client, the client is

Release: Tuesday, April 6, 2010

Name & bitmask	Meaning
0x0008	requesting compressed data for an SMB_COM_READ_ANDX request. If cleared by the server, the server is notifying the client that the data was written uncompressed. This bit field SHOULD only be set to one when NT LAN Manager or later is negotiated for the SMB dialect.
SMB_FLAGS2_SMB_SECURITY_SIGNATURE_REQUIRED 0x0010	This flag MUST be set by the client on the first SMB_COM_SESSION_SETUP_ANDX request (section 2.2.4.6.1) sent to a server that supports extended security if the client requires all further communication with this server to be signed.<20> If the server does not support signing, it MUST disconnect the client by closing the underlying transport connection. Clients and servers MUST ignore this value for other requests and responses. If the client receives a non-signed response from the server, it MUST disconnect the underlying transport connection. This bit field SHOULD only be set to one when NT LAN Manager or later is negotiated for the SMB dialect, the client supports extended security, and the client is configured to require security signatures.
SMB_FLAGS2_IS_LONG_NAME 0x0040	If set, the path contained in the message contains long names; otherwise, the paths are restricted to **8.3 names**. This bit field SHOULD only be set to one when NT LAN Manager or later is negotiated for the SMB dialect.
SMB_FLAGS2_REPARSE_PATH 0x0400	If set, the path in the request MUST contain an @GMT token (that is, a Previous Version token), as specified in section 2.2.1.1.1. This bit field SHOULD be set to one only when NT LAN Manager or later is negotiated for the SMB dialect.
SMB_FLAGS2_EXTENDED_SECURITY 0x0800	Indicates that the client or server supports SPNEGO authentication, as specified in section 3.2.5.2 for client behavior and section 3.3.5.2 for server behavior. This bit field SHOULD be set to one only when NT LAN Manager or later is negotiated for the SMB dialect and the client or server supports extended security.

PIDHigh (2 bytes): This field MUST give the 2 high bytes of the **process identifier (PID)** if the client is using 32-bit process IDs, as specified in [MS-CIFS] section 2.2.3.1. If a client uses 16-bit process IDs, this field MUST be set to zero.<21>

2.2.4   SMB Command Extensions

2.2.4.1   SMB_COM_OPEN_ANDX (0x2D)

2.2.4.1.1   Client Request Extensions

An SMB_COM_OPEN_ANDX request is sent by a client to open a file or named pipe on a server. The new flag value in the **Flags** field of the SMB_COM_OPEN_ANDX request, SMB_OPEN_EXTENDED_RESPONSE, is used to trigger new behavior that is specified in this document. All other fields are as specified in [MS-CIFS] section 2.2.4.41.1.

This command has been deprecated. Client implementations SHOULD use SMB_COM_NT_CREATE_ANDX.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR               AndXCommand;
    UCHAR               AndXReserved;
    USHORT              AndXOffset;
    USHORT              Flags;
    USHORT              AccessMode;
    SMB_FILE_ATTRIBUTES SearchAttrs;
    SMB_FILE_ATTRIBUTES FileAttrs;
    UTIME               CreationTime;
    USHORT              OpenMode;
    ULONG               AllocationSize;
    ULONG               Timeout;
    USHORT              Reserved[2];
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    SMB_STRING FileName;
    }
  }
```

SMB_Parameters

Words (34 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
Flags																AccessMode															

Release: Tuesday, April 6, 2010

SearchAttrs		FileAttrs	
CreationTime			
OpenMode		AllocationSize	
...		Timeout	
...		Reserved	
...			

Flags (2 bytes): A 16-bit field of bit flags. For completeness, all flags are listed in the following table. Bit values listed as reserved SHOULD be set to zero by the client and MUST be ignored by the server.

Name & bitmask	Meaning
SMB_OPEN_QUERY_INFORMATION 0x0001	If set, the client is requesting additional info in the response. The server MUST set **FileDataSize**, **FileAttrs**, **AccessRights**, **ResourceType**, and **NMPipeStatus** in the response. If not set, the server MUST set these fields to zero.
SMB_OPEN_OPLOCK 0x0002	If set, the client is requesting an **oplock**.
SMB_OPEN_OPBATCH 0x0004	If set, the client is requesting a batch oplock.
SMB_OPEN_EXTENDED_RESPONSE 0x0010	If set, the client is requesting the extended format of the response, as described later in this section.
Reserved 0xFFE8	Reserved; SHOULD be set to zero by the client, and MUST be ignored by the server.

2.2.4.1.2   Server Response Extensions

If the client requested extended information by setting SMB_OPEN_EXTENDED_RESPONSE, then a successful response takes the following format. Aside from **WordCount**, **ResourceType**, **ServerFID**, **Reserved**, **MaximalAccessRights**, and **GuestMaximalAccessRights** fields, all other fields are as specified in [MS-CIFS] section 2.2.4.41.2.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR           AndXCommand;
    UCHAR           AndXReserved;
    USHORT          AndXOffset;
    USHORT          FID;
```

```
        SMB_FILE_ATTRIBUTES FileAttrs;
        UTIME               LastWriteTime;
        ULONG               DataSize;
        USHORT              GrantedAccess;
        USHORT              ResourceType;
        USHORT              NMPipeStatus;
        USHORT              OpenResults;
        ULONG               ServerFID;
        USHORT              Reserved;
        ACCESS_MASK         MaximalAccessRights;
        ACCESS_MASK         GuestMaximalAccessRights;
        }
    }
  SMB_Data
    {
    USHORT ByteCount;
    }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SMB_Parameters																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																							SMB_Data								
...																															
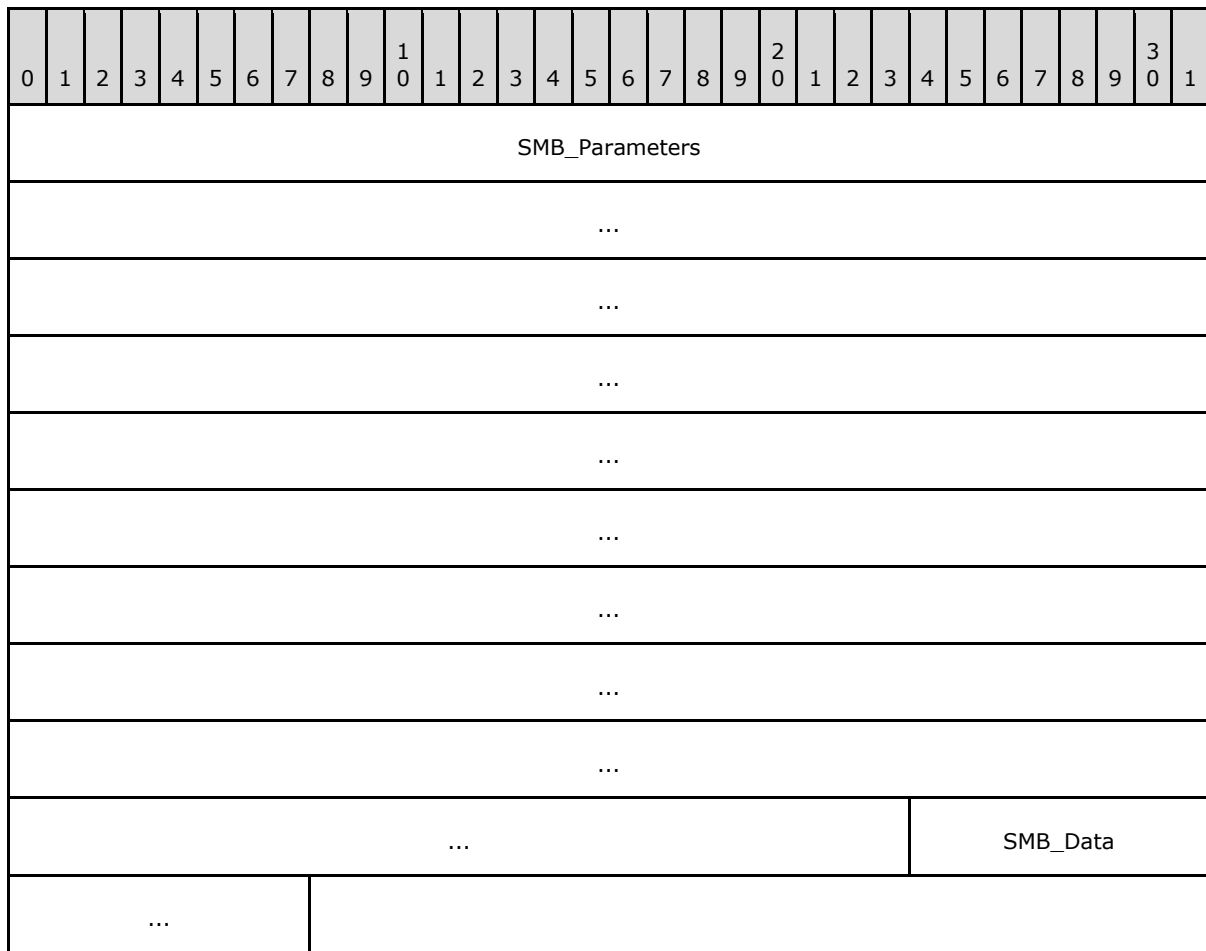
SMB_Parameters (39 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
WordCount								Words																							
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															

WordCount (1 byte): The value of this field MUST be 0x13.

Words (38 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
FID																FileAttrs															
LastWriteTime																															
DataSize																															
GrantedAccess																ResourceType															
NMPipeStatus																OpenResults															

ServerFID	
Reserved	MaximalAccessRights
...	GuestMaximalAccessRights
...	

ResourceType (2 bytes): The file type. This field SHOULD<22> be interpreted as follows:

Name & value	Meaning
FileTypeDisk 0x0000	File or Directory
FileTypeByteModePipe 0x0001	**Byte mode** named pipe
FileTypeMessageModePipe 0x0002	**Message mode** named pipe
FileTypePrinter 0x0003	Printer Device
FileTypeUnknown 0xFFFF	Unknown file type

ServerFID (4 bytes): A 32-bit server file identifier that uniquely identifies the file on the server. This field SHOULD<23> be set to any ULONG value that uniquely identifies the file.

Reserved (2 bytes): An unused value that SHOULD be set to zero when sending this message. The client MUST ignore this field when receiving this message.

MaximalAccessRights (4 bytes): The maximum access rights that this user has on this object. This field MUST be encoded in an ACCESS_MASK format, as specified in section 2.2.1.4.

GuestMaximalAccessRights (4 bytes): The maximum access rights that the **guest account** has on this file. This field MUST be encoded in an ACCESS_MASK format, as specified in section 2.2.1.4. Support and exact specifications of the notion of a guest account is implementation specific. Implementations that do not support the notion of a guest account MUST set this field to zero.<24>

SMB_Data (2 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ByteCount																															

ByteCount (2 bytes): The value of this field SHOULD<25> be set to zero. No data is sent by this message.

2.2.4.2 SMB_COM_READ_ANDX (0x2E)

2.2.4.2.1 Client Request Extensions

An SMB_COM_READ_ANDX request is sent by a client to read from a file or named pipe on a server. These extensions overload the **Timeout** field with the new **Timeout_or_MaxCountHigh** field, which allows the use of read lengths above 0xFFFF when CAP_LARGE_READX has been negotiated. All other fields are defined as specified in [MS-CIFS] section 2.2.4.42.1.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT FID;
    ULONG  Offset;
    USHORT MaxCountOfBytesToReturn;
    USHORT MinCountOfBytesToReturn;
    ULONG  Timeout_or_MaxCountHigh;
    USHORT Remaining;
    ULONG  OffsetHigh (optional);
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  }
```

SMB_Parameters

Words (24 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset																
FID																Offset																
...																MaxCountOfBytesToReturn																
MinCountOfBytesToReturn																Timeout_or_MaxCountHigh																
...																Remaining																
OffsetHigh (optional)																																

Timeout_or_MaxCountHigh (4 bytes): This field is extended to be treated as a union of a 32-bit **Timeout** field and a 16-bit **MaxCountHigh** field. For read operations on named pipes, this field MUST be used and interpreted as the 32-bit **Timeout** field and MUST be set to 0xFFFFFFFF:

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
															Timeout																
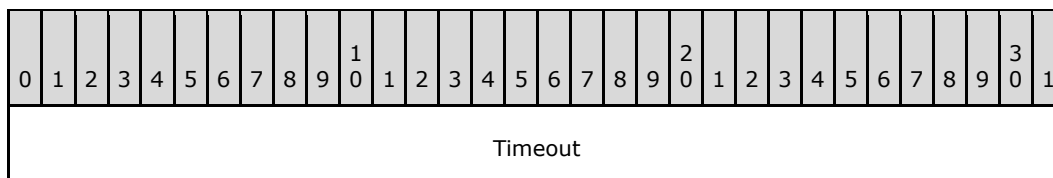
Timeout (4 bytes): The client can set the **Timeout** field to a requested time-out value in milliseconds or the client can set the **Timeout** field to 0xFFFFFFFF, which indicates that the operation MUST NOT time out. For read operations on named pipes this field MUST be set to 0xFFFFFFFF.

For read operations on files, this field MUST be used and interpreted as the 16-bit **MaxCountHigh** field followed by a 16-bit **Reserved** field:

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
						MaxCountHigh														Reserved											
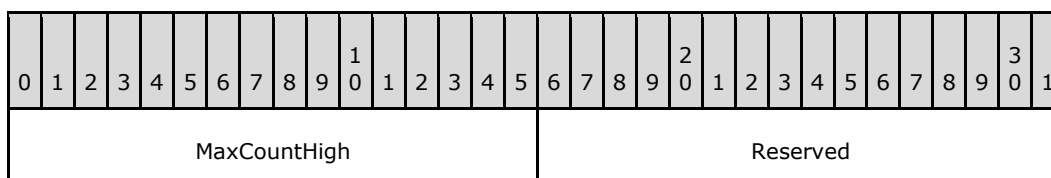
MaxCountHigh (2 bytes): This field is a 16-bit integer. If the read being requested is larger than 0xFFFF bytes in size, then the client MUST use the **MaxCountHigh** field to hold the two most significant bytes of the requested size, which allows for 32-bit read lengths to be requested when combined with **MaxCountOfBytesToReturn**. If the read is not larger than 0xFFFF bytes, then the client MUST set the **MaxCountHigh** to zero.<26>

Reserved (2 bytes): Unlike most other reserved fields, this field can sometimes take specific values. The **Reserved** field SHOULD be set to zero by the client if **MaxCountHigh** is zero, and SHOULD be set to 0xFFFF by the client if **MaxCountHigh** is 0xFFFF. For all other values, this field SHOULD be set to zero by the client. For all values, this field MUST be ignored by the server.

2.2.4.2.2 Server Response Extensions

A successful response takes the following format. Aside from the first two bytes of the **SMB_Parameters.Words.Reserved2[]** field being extended for use as the new **DataLengthHigh** field, all other fields are defined as specified in [MS-CIFS] section 2.2.4.42.2.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT Available;
    USHORT DataCompactionMode;
```

```
    USHORT Reserved1;
    USHORT DataLength;
    USHORT DataOffset;
    USHORT DataLengthHigh;
    USHORT Reserved2[4];
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR Pad[] (optional);
    UCHAR Data[DataLength & DataLengthHigh];
    }
  }
```

SMB_Parameters

Words (24 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
Available																DataCompactionMode															
Reserved1																DataLength															
DataOffset																DataLengthHigh															
Reserved2																															
...																															

DataLengthHigh (2 bytes): If the data read is greater than or equal to 0x00010000 bytes (64KB) in length, then the server MUST set the two least-significant bytes of the length in the **DataLength** field of the response and the two most-significant bytes of the length in the **DataLengthHigh** field. Otherwise, this field MUST be set to zero.

Reserved2 (8 bytes): This field MUST be set to zero by the server and MUST be ignored by the client. The last 4 words are reserved in order to make the SMB_COM_READ_ANDX response the same size as the SMB_COM_WRITE_ANDX response.

2.2.4.3 SMB_COM_WRITE_ANDX (0x2F)

2.2.4.3.1 Client Request Extensions

An SMB_COM_WRITE_ANDX request is sent by a client to write data to a file or named pipe on a server. These extensions allocate the **SMB_Parameters.Words.Reserved** field for use as the **DataLengthHigh** field. This field is used when the CAP_LARGE_WRITEX capability has been negotiated to allow for file writes larger than 0xFFFF bytes in length. All other fields are defined as specified in [MS-CIFS] section 2.2.4.43.1.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT FID;
    ULONG  Offset;
    ULONG  Timeout;
    USHORT WriteMode;
    USHORT Remaining;
    USHORT DataLengthHigh;
    USHORT DataLength;
    USHORT DataOffset;
    ULONG  OffsetHigh (optional);
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR Pad;
    UCHAR Data[DataLength & DataLengthHigh];
    }
  }
```

SMB_Parameters

Words (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
FID																Offset															
...																Timeout															

...	WriteMode
Remaining	DataLengthHigh
DataLength	DataOffset
OffsetHigh (optional)	

DataLengthHigh (2 bytes): This field contains the two most significant bytes of the length of the data to write to the file. If the number of bytes to be written is greater than or equal to 0x00010000(64 kilobytes), then the client MUST set the two least significant bytes of the length in the **DataLength** field of the request and the two most significant bytes of the length in the **DataLengthHigh** field.

2.2.4.3.2 Server Response Extensions

A successful response takes the following format. These extensions allocate the first two bytes of the **SMB_Parameters.Words.Reserved** field for use as the **CountHigh** field. This field is used when the CAP_LARGE_WRITEX capability has been negotiated to allow for file writes larger than 0xFFFF bytes in length. All other fields are defined as specified in [MS-CIFS] section 2.2.4.43.2.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT Count;
    USHORT Available;
    USHORT CountHigh;
    USHORT Reserved;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  }
```

SMB_Parameters

Words (12 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															

Count	Available
CountHigh	Reserved

CountHigh (2 bytes): This field contains the two most significant bytes of the count of bytes written. If the number of bytes written is greater than or equal to 0x00010000(64 kilobytes), then the server MUST set the two least significant bytes of the length in the **Count** field of the request and the two most significant bytes of the length in the **CountHigh** field.

Reserved (2 bytes): This field is reserved. Servers MUST set this field to zero and clients MUST ignore this field upon receipt.

2.2.4.4   SMB_COM_TRANSACTION2 (0x32) Extensions

The SMB_COM_TRANSACTION2 request is sent by a client to execute a specific operation of various types on the server. These operations include file enumeration, query and set file attribute operations, and DFS referral retrieval. The general format of the SMB_COM_TRANSACTION2 command requests and responses is given in [MS-CIFS] sections 2.2.4.46 and 2.2.4.47. Execution of SMB_COM_TRANSACTION2 is defined as specified in [MS-CIFS] sections 3.2.4.1.5, 3.2.5.1.4, and 3.3.5.1.5.

Valid SMB_COM_TRANSACTION2 subcommand codes, also known as "Trans2 subcommands", are specified in section 2.2.2.2. The format and syntax of these subcommands are specified in section 2.2.6 and in [MS-CIFS] section 2.2.6.

2.2.4.5   SMB_COM_NEGOTIATE (0x72)

2.2.4.5.1   Client Request Extensions

All fields are defined as specified in [MS-CIFS] section 2.2.4.52.1. In order to support the extension in this document, the client MUST include the NT LAN Manager dialect (identified by the "NT LM 0.12" dialect string) in the **SMB_Data.Bytes.Dialects[]** array of the request.

When set, the **SMB_Header.Flags2** SMB_FLAGS2_EXTENDED_SECURITY flag indicates support for specification [RFC2743] and GSS authentication (see section 3.1.5.1), and indicates to the server that it should send an Extended Security response (see section 2.2.4.5.2).

2.2.4.5.2   Server Response Extensions

2.2.4.5.2.1   Extended Security Response

If the selected dialect is NT LAN Manager and the client has indicated extended security is being used, a successful response MUST take the following form. Aside from the additional notes to the **SMB_Parameters.Words.MaxBufferSize** and **SMB_Parameters.Words.ChallengeLength** fields, the new **SMB_Parameters.Words.Capabilities** bits, and the **SMB_Data.Bytes.ServerGuid** and **SMB_Data.Bytes.SecurityBlob** fields, all other fields are defined as specified in [MS-CIFS] section 2.2.4.52.2.

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
```

```
    {
    USHORT    DialectIndex;
    UCHAR     SecurityMode;
    USHORT    MaxMpxCount;
    USHORT    MaxNumberVcs;
    ULONG     MaxBufferSize;
    ULONG     MaxRawSize;
    ULONG     SessionKey;
    ULONG     Capabilities;
    FILETIME  SystemTime;
    SHORT     ServerTimeZone;
    UCHAR     ChallengeLength;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    GUID ServerGUID;
    UCHAR  SecurityBlob[];
    }
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SMB_Parameters																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																								SMB_Data (variable)							
...																															

SMB_Parameters (35 bytes):

Release: Tuesday, April 6, 2010

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
WordCount								Words																							
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															

Words (34 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
DialectIndex																SecurityMode								MaxMpxCount							
...								MaxNumberVcs																MaxBufferSize							
...																								MaxRawSize							
...																								SessionKey							
...																								Capabilities							
...																								SystemTime							
...																															
...																								ServerTimeZone							

...	ChallengeLength

MaxBufferSize (4 bytes): Maximum size, in bytes, of the server buffer for receiving SMB messages. This value accounts for the size of the largest SMB message that the client can send to the server, measured from the start of the SMB header to the end of the packet. This value does not account for any underlying transport-layer packet headers, and thus does not account for the size of the complete network packet.<27>

The only exceptions in which this maximum buffer size MUST be exceeded are:

- When the SMB_COM_WRITE_ANDX command is used and the client and server both support the CAP_LARGE_WRITEX capability (see the **Capabilities** field for more information).

- When the SMB_COM_WRITE_RAW command is used and both the client and server support the CAP_RAW_MODE capability.

Capabilities (4 bytes): A 32-bit field providing a set of server capability indicators. This bit field is used to indicate to the client which features are supported by the server. Any value not listed in the following table is unused. The server MUST set the unused bits to zero. The client MUST ignore these bits.

These extensions provide the following new capability bits:

- CAP_COMPRESSED_DATA

- CAP_DYNAMIC_REAUTH

- CAP_EXTENDED_SECURITY

- CAP_INFOLEVEL_PASSTHRU

- CAP_LARGE_WRITEX

- CAP_LWIO

- CAP_UNIX

The rest of the values in the capabilities table are included for completeness.

Name & bitmask	Meaning
CAP_RAW_MODE 0x00000001	The server supports SMB_COM_READ_RAW and SMB_COM_WRITE_RAW requests.<28> Raw mode is not supported over connectionless transports.
CAP_MPX_MODE 0x00000002	The server supports SMB_COM_READ_MPX and SMB_COM_WRITE_MPX requests.<29> MPX mode is supported only over connectionless transports.
CAP_UNICODE 0x00000004	The server supports UTF-16LE **Unicode strings**.
CAP_LARGE_FILES 0x00000008	The server supports large files with 64-bit offsets.

Name & bitmask	Meaning
CAP_NT_SMBS 0x00000010	The server supports SMB commands particular to the NT LAN Manager dialect.
CAP_RPC_REMOTE_APIS 0x00000020	The server supports the use of remote procedure call [MS-RPCE] for remote API calls. Similar functionality would otherwise require use of the legacy Remote Administration Protocol, as specified in [MS-RAP].
CAP_STATUS32 0x00000040	The server is capable of responding with 32-bit status codes in the **Status** field of the SMB header (for more information, see [MS-CIFS] 2.2.3.1). CAP_STATUS32 can also be referred to as CAP_NT_STATUS.
CAP_LEVEL_II_OPLOCKS 0x00000080	The server supports level II opportunistic locks (oplocks).
CAP_LOCK_AND_READ 0x00000100	The server supports the SMB_COM_LOCK_AND_READ command requests.
CAP_NT_FIND 0x00000200	The server supports the TRANS2_FIND_FIRST2, TRANS2_FIND_NEXT2, and FIND_CLOSE2 command requests. This bit SHOULD<30> be set if CAP_NT_SMBS is set.
CAP_DFS 0x00001000	The server is aware of the DFS Referral Protocol, as specified in [MS-DFSC], and can respond to DFS referral requests. For more information, see [MS-CIFS] sections 2.2.6.16.1 and 2.2.6.16.2.
CAP_INFOLEVEL_PASSTHRU 0x00002000	The server supports pass-through Information Levels, as specified in section 2.2.2.3. This allows the client to pass Information Level structures in QUERY and SET operations.<31>
CAP_LARGE_READX 0x00004000	The server supports large read operations. This capability affects the maximum size, in bytes, of the server buffer for sending an SMB_COM_READ_ANDX response to the client. When this capability is set by the server (and set by the client in the SMB_COM_SESSION_SETUP_ANDX request), then the maximum server buffer size for sending data can exceed the **MaxBufferSize** field. Therefore, the server can send a single SMB_COM_READ_ANDX response to the client up to this size.<32> When signing is active on a connection, then clients MUST limit read lengths to the **MaxBufferSize** value negotiated by the server irrespective of the value of the CAP_LARGE_READX flag.
CAP_LARGE_WRITEX 0x00008000	The server supports large write operations. This capability affects the maximum size, in bytes, of the server buffer for receiving an SMB_COM_WRITE_ANDX client request. When this capability is set by the server (and set by the client in the SMB_COM_SESSION_SETUP_ANDX request), then the maximum server buffer size of bytes it writes can exceed the **MaxBufferSize** field. Therefore, a client

Name & bitmask	Meaning
	can send a single SMB_COM_WRITE_ANDX request up to this size.<33>
When signing is active on a connection, then clients MUST limit write lengths to the **MaxBufferSize** value negotiated by the server, irrespective of the value of the CAP_LARGE_WRITEX flag.
CAP_LWIO
0x00010000	The server supports new light-weight I/O control (**IOCTL**) and file system control (FSCTL) operations. These operations are accessed using the NT_TRANSACT_IOCTL subcommand (section 2.2.7.2).<34>
CAP_UNIX
0x00800000	The server supports UNIX extensions.<35> For more information, see [SNIA].
CAP_COMPRESSED_DATA
0x02000000	Reserved but not implemented.<36>
The server supports compressed SMB packets.
CAP_DYNAMIC_REAUTH
0x20000000	The server supports re-authentication, if required, as specified in sections 3.3.5.1 and 3.2.5.1.<37>
CAP_PERSISTENT_HANDLES
0x40000000	Reserved but not implemented.<38>
The server supports persistent handles.
CAP_EXTENDED_SECURITY
0x80000000	The server supports extended security for authentication, as specified in section 3.2.4.2.4. This bit is used in conjunction with the SMB_FLAGS2_EXTENDED_SECURITY **SMB_Header.Flags2** flag, as specified in section 2.2.3.1.

ChallengeLength (1 byte): When the CAP_EXTENDED_SECURITY bit is set, the server MUST set this value to zero and clients MUST ignore this value.

   SMB_Data (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ByteCount																Bytes (variable)															
...																															

   ByteCount (2 bytes): This field MUST be greater than or equal to 0x0010.

   Bytes (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ServerGUID																															
...																															
...																															
...																															
SecurityBlob (variable)																															
...																															

ServerGUID (16 bytes): This field MUST be a GUID generated by the server to uniquely identify this server. This field SHOULD NOT be used by a client as a secure method of identifying a server because it can be forged. A client SHOULD use this information to detect whether connections to different textual names resolve to the same target server when direct TCP is used. This knowledge can then be used to appropriately set the **SMB_Parameters.Words.VcNumber** field in the SMB_COM_SESSION_SETUP_ANDX request (see [MS-CIFS] section 2.2.4.53.1).<39>

SecurityBlob (variable): A security binary large object (BLOB) that SHOULD contain an authentication token as produced by the GSS protocol (as specified in section 3.2.4.2.4 and [RFC2743]).<40>

2.2.4.5.2.2 Non-Extended Security Response

If extended security is not being used and the NT LAN Manager dialect has been selected, then a successful response MUST take the following form. Aside from the new **SMB_Parameters.Words.Capabilities** bits, the additional notes to the **SMB_Parameters.Words.MaxBufferSize** field, and the **SMB_Data.Bytes.ServerName** field, all other fields are defined as specified in [MS-CIFS] section 2.2.4.52.2. The **SMB_Parameters.Words.ChallengeLength** field and the entire **SMB_Data** block are included from [MS-CIFS] to highlight the differences between the Extended and Non-Extended Security responses.

In order to determine whether the **SMB_Data.Bytes.ServerName** field is present, the client MUST check the **SMB_Data.ByteCount** field to determine whether additional data is present beyond the NULL terminator of the **SMB_Data.Bytes.DomainName** string.

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    USHORT   DialectIndex;
    UCHAR    SecurityMode;
    USHORT   MaxMpxCount;
    USHORT   MaxNumberVcs;
```

Release: Tuesday, April 6, 2010

```
              ULONG    MaxBufferSize;
              ULONG    MaxRawSize;
              ULONG    SessionKey;
              ULONG    Capabilities;
              FILETIME SystemTime;
              SHORT    ServerTimeZone;
              UCHAR    ChallengeLength;
              }
          }
       SMB_Data
          {
          USHORT ByteCount;
          Bytes
             {
             UCHAR       Challenge[];
             SMB_STRING DomainName[];
             SMB_STRING ServerName[];
             }
          }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SMB_Parameters																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																								SMB_Data (variable)							
...																															

SMB_Parameters (35 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Words																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															

Words (34 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
DialectIndex																SecurityMode								MaxMpxCount							
...								MaxNumberVcs																MaxBufferSize							
...																MaxRawSize															
...																SessionKey															
...																Capabilities															
...																SystemTime															
...																																
...																ServerTimeZone															

...	ChallengeLength

MaxBufferSize (4 bytes): Maximum size, in bytes, of the server buffer for receiving SMB messages. This value accounts for the size of the largest SMB message that the client can send to the server, measured from the start of the SMB header to the end of the packet. This value does not account for any underlying transport-layer packet headers and thus does not account for the size of the complete network packet.<41>

The only exceptions in which this maximum buffer size MUST be exceeded are:

- When the SMB_COM_WRITE_ANDX command is used and both the client and server support the CAP_LARGE_WRITEX capability (see the **Capabilities** field for more information).

- When the SMB_COM_WRITE_RAW command is used and both the client and server support the CAP_RAW_MODE capability.

Capabilities (4 bytes): A 32-bit field providing a set of server capability indicators. This bit field is used to indicate to the client which features are supported by the server. Any value not listed in the following table is unused. The server MUST set the unused bits to zero in a response and the client MUST ignore these bits.

There are several new capability bits:

- CAP_COMPRESSED_DATA

- CAP_DYNAMIC_REAUTH

- CAP_EXTENDED_SECURITY

- CAP_INFOLEVEL_PASSTHRU

- CAP_LARGE_WRITEX

- CAP_LWIO

- CAP_UNIX

Any value not listed in the following table SHOULD be unused. A server SHOULD set the unused bits to zero in a response and a client MUST ignore these bits. The table of server capabilities is provided in the previous section.

ChallengeLength (1 byte): This field MUST be either 0x00 or 0x08 and is the length of the random challenge used in challenge/response authentication. If the server does not support challenge/response authentication, then this field MUST be 0x00. This field is often referred to as EncryptionKeyLength.

SMB_Data (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ByteCount																Bytes (variable)															

ByteCount (2 bytes): This field MUST be greater than or equal to 0x0003.

Bytes (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Challenge (variable)																															
...																															
DomainName (variable)																															
...																															
ServerName (variable)																															
...																															

Challenge (variable): An array of unsigned bytes that MUST be the length of the number of bytes specified in the **ChallengeLength** field and MUST represent the server challenge. This array MUST NOT be NULL-terminated.<42>

DomainName (variable): The name of the **domain** or workgroup to which the server belongs.

ServerName (variable): A variable-length, NULL-terminated Unicode string that contains the name of the Server.

2.2.4.6   SMB_COM_SESSION_SETUP_ANDX (0x73)

2.2.4.6.1   Client Request Extensions

An SMB_COM_SESSION_SETUP_ANDX request MUST be sent by a client to begin user authentication on an SMB connection and establish an SMB session.

When extended security is being used (see section 3.2.4.2.4), the request MUST take the following form. Aside from the **SecurityBlobLength** field, the additional capabilities used in the **Capabilities** field, and the **ByteCount** and **SecurityBlob** fields, all other fields are as specified in [MS-CIFS] section 2.2.4.53.1.

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
```

```
     USHORT AndXOffset;
     USHORT MaxBufferSize;
     USHORT MaxMpxCount;
     USHORT VcNumber;
     ULONG  SessionKey;
     USHORT SecurityBlobLength;
     ULONG  Reserved;
     ULONG  Capabilities;
     }
   }
 SMB_Data
   {
   USHORT ByteCount;
   Bytes
     {
     UCHAR       SecurityBlob[SecurityBlobLength];
     SMB_STRING NativeOS[];
     SMB_STRING NativeLanMan[];
     }
   }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SMB_Parameters																																
...																																
...																																
...																																
...																																
...																																
...								SMB_Data (variable)																								
...																																

SMB_Parameters (25 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
WordCount								Words																								

53 / 180

[MS-SMB] — v20100406
Server Message Block (SMB) Protocol Specification

Copyright © 2010 Microsoft Corporation.

Release: Tuesday, April 6, 2010

...
...
...
...
...

...

WordCount (1 byte): The value of this field MUST be 0x0C.

Words (24 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
MaxBufferSize																MaxMpxCount															
VcNumber																SessionKey															
...																SecurityBlobLength															
Reserved																															
Capabilities																															

SecurityBlobLength (2 bytes): This value MUST specify the length in bytes of the variable-length **SecurityBlob** field that is contained within the request.

Capabilities (4 bytes): A set of client capabilities. This field has the same structure as the **SMB_Parameters.Capabilities** field of the SMB_COM_NEGOTIATE Server Response specified in section 2.2.4.5.2.<43>Servers MUST NOT check for Kerberos ticket expiry if the client does not support the CAP_DYNAMIC_REAUTH capability.<44>

SMB_Data (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ByteCount																Bytes (variable)															
...																															

ByteCount (2 bytes): If SMB_FLAGS2_UNICODE is set in the **SMB_Header.Flags2** field, then this field MUST be greater than or equal to 0x0004. If SMB_FLAGS2_UNICODE is not set, then this field MUST be greater than or equal to 0x0002.

Bytes (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SecurityBlob (variable)																															
...																															
NativeOS (variable)																															
...																															
NativeLanMan (variable)																															
...																															

SecurityBlob (variable): This field MUST be the authentication token sent to the server, as specified in section 3.2.4.2.4 and in [RFC2743].

NativeOS (variable): A string that represents the native operating system of the SMB client. If SMB_FLAGS2_UNICODE is set in the **Flags2** field of the SMB header of the request, then the name string MUST be a NULL-terminated array of 16-bit Unicode characters. Otherwise, the name string MUST be a NULL-terminated array of **OEM character**s. If the name string consists of Unicode characters, then this field MUST be aligned to start on a 2-byte boundary from the start of the SMB header.<45>

NativeLanMan (variable): A string that represents the native LAN manager type of the client. If SMB_FLAGS2_UNICODE is set in the **Flags2** field of the SMB header of the request, then the name string MUST be a NULL-terminated array of 16-bit Unicode characters. Otherwise, the name string MUST be a NULL-terminated array of OEM characters. If the name string consists of Unicode characters, then this field MUST be aligned to start on a 2-byte boundary from the start of the SMB header.<46>
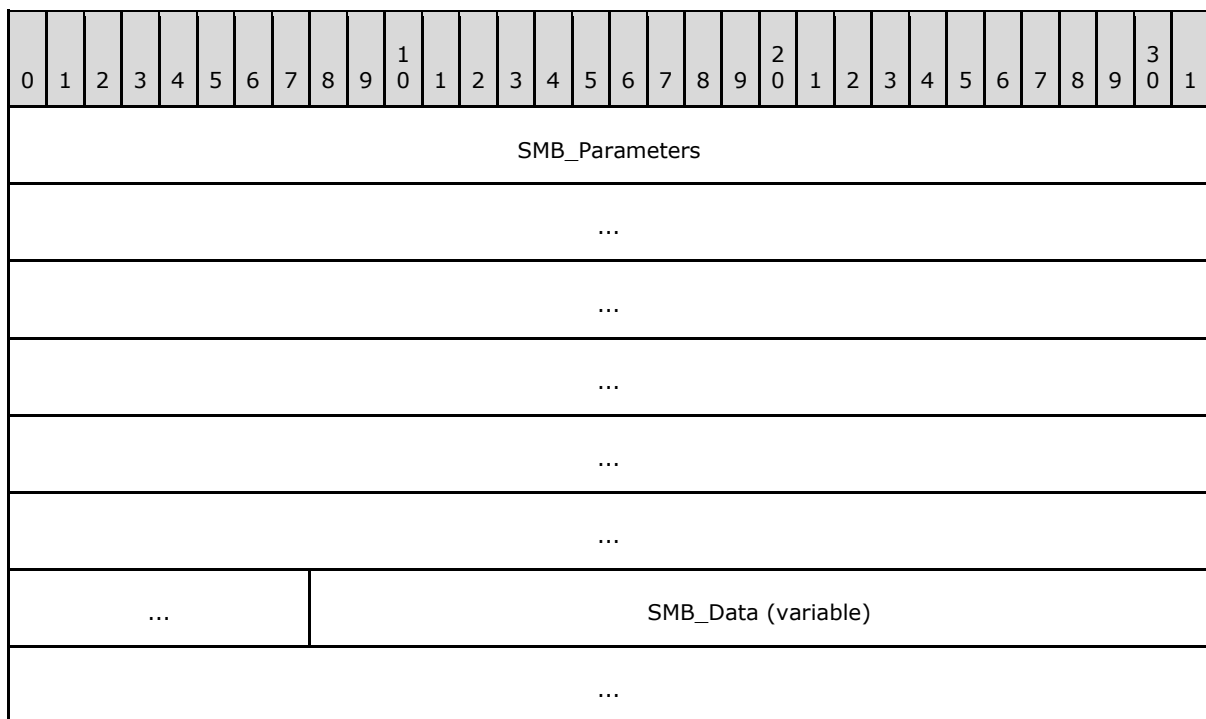
2.2.4.6.2 Server Response Extensions

When extended security is being used (see section 3.2.4.2.4), a successful response MUST take the following form. Aside from the SecurityBlobLength field, the additional capabilities used in the Capabilities field, and the ByteCount and SecurityBlob fields, all of the other fields are as specified in [MS-CIFS] section 2.2.4.53.2.

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT Action;
    USHORT SecurityBlobLength;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR      SecurityBlob[SecurityBlobLength];
    SMB_STRING NativeOS;
    SMB_STRING NativeLanMan;
    SMB_STRING PrimaryDomain;
    }
  }
```
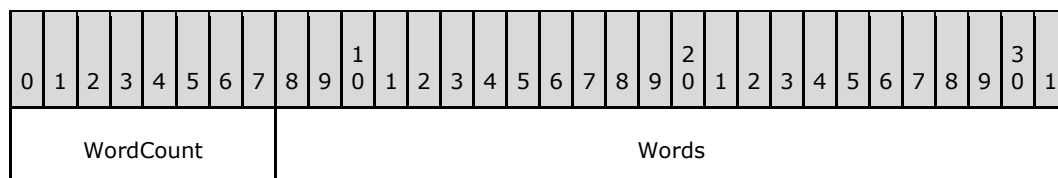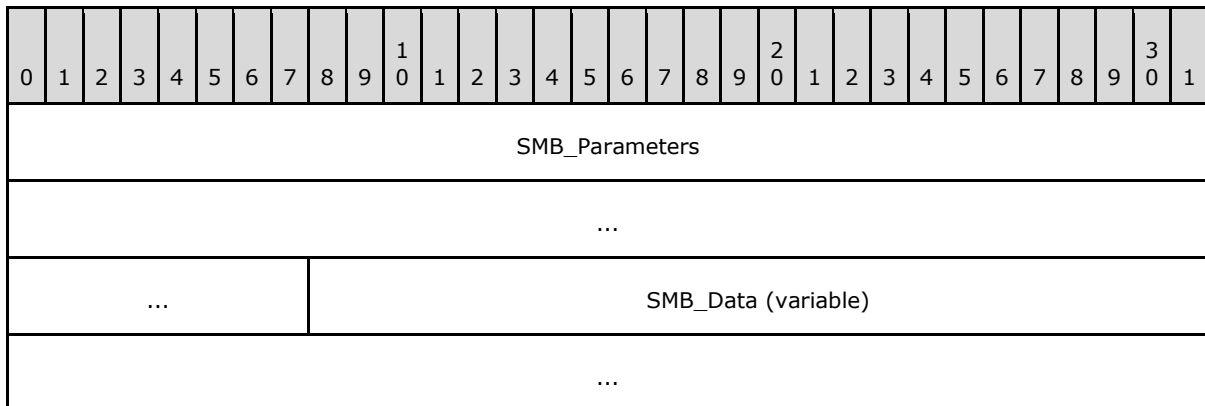
0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SMB_Parameters																															
...																															
...							SMB_Data (variable)																								
...																															

SMB_Parameters (9 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
WordCount								Words																							
...																															

Release: Tuesday, April 6, 2010
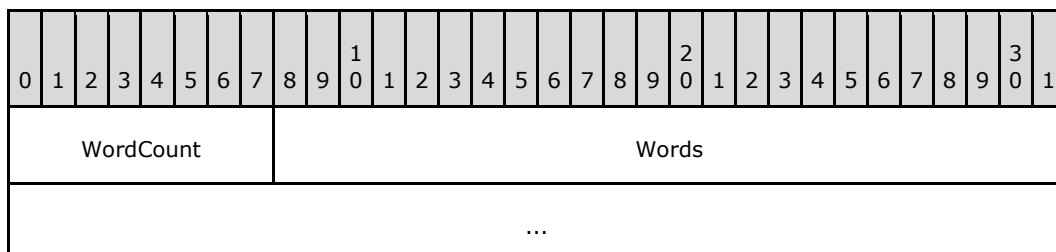
WordCount (1 byte): The value of this field MUST be 0x04.

Words (8 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
Action																SecurityBlobLength															

Action (2 bytes): A 16-bit field. The two lowest-order bits have been defined.

Name and bitmask	Meaning
SMB_SETUP_GUEST 0x0001	If clear (0x0000), then the user successfully authenticated and is logged in. If set (0x0001), then authentication failed but the server has granted guest access; the user is logged in as a Guest.
SMB_SETUP_USE_LANMAN_KEY 0x0002	This bit is not used with extended security and MUST be clear (0x0000).

The server's response does not specify whether the access granted is of type Anonymous. However, the security system can provide that information once authorization completes.

SecurityBlobLength (2 bytes): This value MUST specify the length, in bytes, of the variable-length **SecurityBlob** that is contained within the response.

SMB_Data (variable):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ByteCount																Bytes (variable)															
...																															

ByteCount (2 bytes): If SMB_FLAGS2_UNICODE is set in the **SMB_Header.Flags2** field, then this field MUST be greater than or equal to 0x0006. If SMB_FLAGS2_UNICODE is not set, then this field MUST be greater than or equal to 0x0003.

Bytes (variable):

	0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SecurityBlob (variable)																																	
...																																	
NativeOS (variable)																																	
...																																	
NativeLanMan (variable)																																	
...																																	
PrimaryDomain (variable)																																	
...																																	

SecurityBlob (variable): This value MUST contain the authentication token being returned to the client, as specified in section 3.3.5.3 and [RFC2743].

NativeOS (variable): A string that represents the native operating system of the server. If SMB_FLAGS2_UNICODE is set in the **Flags2** field of the SMB header of the response, then the string MUST be a NULL-terminated array of 16-bit Unicode characters. Otherwise, the string MUST be a NULL-terminated array of OEM characters. If the name string consists of Unicode characters, then this field MUST be aligned to start on a 2-byte boundary from the start of the SMB header.<47>

NativeLanMan (variable): A string that represents the native LAN Manager type of the server. If SMB_FLAGS2_UNICODE is set in the **Flags2** field of the SMB header of the response, then the string MUST be a NULL-terminated array of 16-bit Unicode characters. Otherwise, the string MUST be a NULL-terminated array of OEM characters. If the name string consists of Unicode characters, then this field MUST be aligned to start on a 2-byte boundary from the start of the SMB header.<48>

PrimaryDomain (variable): The value of the **PrimaryDomain** field represents the primary domain or workgroup name of the server. If SMB_FLAGS2_UNICODE is set in the **Flags2** field of the SMB header of the response, then the string MUST be a NULL-terminated array of 16-bit Unicode characters. Otherwise, the string MUST be a NULL-terminated array of OEM characters. If the string consists of Unicode characters, then this field MUST be aligned to start on a 2-byte boundary from the start of the SMB header.

2.2.4.7 SMB_COM_TREE_CONNECT_ANDX (0x75)

2.2.4.7.1 Client Request Extensions

An SMB_COM_TREE_CONNECT_ANDX request MUST be sent by a client to establish a **tree connect** to a share. These extensions define new flags (TREE_CONNECT_ANDX_EXTENDED_SIGNATURES and TREE_CONNECT_ANDX_EXTENDED_RESPONSE) in the **SMB_Parameters.Words.Flags** field that are used to trigger the new behavior defined in this specification. The full field description from

[MS-CIFS] is included for completeness. All other fields are as specified in [MS-CIFS] section 2.2.4.55.1.

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR  AndXCommand;
    UCHAR  AndXReserved;
    USHORT AndXOffset;
    USHORT Flags;
    USHORT PasswordLength;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    UCHAR      Password[PasswordLength];
    UCHAR      Pad[];
    SMB_STRING Path;
    OEM_STRING Service;
    }
  }
```

SMB_Parameters

Words (8 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
Flags																PasswordLength															

Flags (2 bytes): A set of options that modify the SMB_COM_TREE_CONNECT_ANDX request. The entire flag set is given here with its symbolic constants. Any combination of the following flags is valid. Any values not given as follows are considered reserved. The client MUST set them to zero, and the server MUST ignore them.

Name & bitmask	Meaning
TREE_CONNECT_ANDX_DISCONNECT_TID 0x0001	If set, the tree connect specified by the TID in the SMB header of the request SHOULD be disconnected when the server sends the response. If this tree disconnect fails, then the error SHOULD be ignored.
TREE_CONNECT_ANDX_EXTENDED_SIGNATURES 0x0004	If set, then the client is requesting signing key protection, as specified in sections 3.2.4.2.5 and 3.2.5.4.

Name & bitmask	Meaning
TREE_CONNECT_ANDX_EXTENDED_RESPONSE 0x0008	If set, then the client is requesting extended information in the SMB_COM_TREE_CONNECT_ANDX response.

2.2.4.7.2 Server Response Extensions

When a server returns extended information, the response takes the following format. Aside from the **WordCount**, **MaximalShareAccessRights**, and **GuestMaximalShareAccessRights** fields, and the new **OptionalSupport** flags, all other fields are defined as specified in [MS-CIFS] section 2.2.4.55.2.

```
SMB_Parameters
  {
  UCHAR  WordCount;
  Words
    {
    UCHAR        AndXCommand;
    UCHAR        AndXReserved;
    USHORT       AndXOffset;
    USHORT       OptionalSupport;
    ACCESS_MASK MaximalShareAccessRights;
    ACCESS_MASK GuestMaximalShareAccessRights;
    }
  }
SMB_Data
  {
  USHORT ByteCount;
  Bytes
    {
    OEM_STRING Service;
    SMB_STRING NativeFileSystem;
    }
  }
```

SMB_Parameters

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
WordCount								Words																							
...																															
...																															
...																															

WordCount (1 byte): The value of this field MUST be 0x07.

Words (14 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
OptionalSupport																MaximalShareAccessRights															
...																GuestMaximalShareAccessRights															
...																															

OptionalSupport (2 bytes): The following **OptionalSupport** bit fields are new extensions: SMB_CSC_MASK, SMB_UNIQUE_FILE_NAME, and SMB_EXTENDED_SIGNATURES. Values from [MS-CIFS] are included for completeness. The values of SMB_CSC_MASK each have their own name and are included for reference purposes only. Any combination of the following flags MUST be supported. All undefined values are considered reserved. The server SHOULD set them to zero and the client MUST ignore them.

Name & bitmask	Value	Meaning
SMB_SUPPORT_SEARCH_BITS 0x0001	0	If set, then the server supports the use of SMB_FILE_ATTRIBUTES Search Attributes in client directory search requests, as specified in [MS-CIFS].
	1	
SMB_SHARE_IS_IN_DFS 0x0002	0	If set, this share is managed by Distributed File System (DFS), as specified in [MS-DFSC].
	1	
SMB_CSC_MASK 0x000C	0	SMB_CSC_CACHE_MANUAL_REINT Clients are allowed to cache files that the user requests for offline use, but there is no automatic file-by-file reintegration.
	1	SMB_CSC_CACHE_AUTO_REINT Clients are allowed to automatically cache the files that a user or application modifies for offline use. Automatic file-by-file reintegration MUST be permitted.
	2	SMB_CSC_CACHE_VDO Clients are allowed to automatically cache the files that a user or application modifies for offline use. Clients are permitted to work from their local cache even while online.
	3	SMB_CSC_NO_CACHING No offline caching is allowed for this share.
SMB_UNIQUE_FILE_NAME	0	If set, then the server is using long file names

Name & bitmask	Value	Meaning
0x0010	1	only and does not support short file names. If set, then the server allows the client to assume that there is no name aliasing for this share (in other words, a single file cannot have two different names). If set, then the server permits the client to cache directory enumerations and file metadata based on the pathname. The client MAY<49> choose to satisfy file attribute queries from its cache and thus could present a slightly stale view of files on the share. The client MUST NOT cache remote file system information for more than 60 seconds.
SMB_EXTENDED_SIGNATURES 0x0020	0	If set, then the server is using signing key protection (see section 3.3.5.4), as requested by the client.
	1	

MaximalShareAccessRights (4 bytes): This field MUST specify the maximum rights that the user has to this share based on the security enforced by the share. This field MUST be encoded in an ACCESS_MASK format, as specified in section 2.2.1.4.

GuestMaximalShareAccessRights (4 bytes): This field MUST specify the maximum rights that the guest account has on this share based on the security enforced by the share. Note that the notion of a guest account is implementation specific.<50>

Implementations that do not support the notion of a guest account MUST set this field to zero, which implies no access. This field MUST be encoded in an ACCESS_MASK format, as specified in section 2.2.1.4.

2.2.4.8   SMB_COM_NT_TRANSACT (0xA0) Extensions

The SMB_COM_NT_TRANSACT request is sent by a client to specify operations on the server. The operations include file open, file create, device I/O control, notify directory change, and set and query security descriptors. The general format of the SMB_COM_NT_TRANSACT command requests and responses is given in [MS-CIFS] sections 2.2.4.62 and 2.2.4.63. Execution of SMB_COM_NT_TRANSACT is as specified in [MS-CIFS] sections 3.2.4.1.5, 3.2.5.1.4, and 3.3.5.1.5.

Valid SMB_COM_NT_TRANSACT subcommand codes, also known as "NT Trans subcommand" codes, are specified in section 2.2.2.2. The format and syntax of these subcommands are specified in section 2.2.7 and in [MS-CIFS] section 2.2.7.

2.2.4.9   SMB_COM_NT_CREATE_ANDX (0xA2)

2.2.4.9.1   Client Request Extensions

An SMB_COM_NT_CREATE_ANDX request is sent by a client to open a file or device on the server. This extension adds the following:

- An additional flag bit is added to the **Flags** field of the SMB_COM_NT_CREATE_ANDX request. The additional flag, NT_CREATE_REQUEST_EXTENDED_RESPONSE, is used to request an extended response from the server.

▪ An additional parameter value is added to the **ImpersonationLevel** field. SECURITY_DELEGATION is added to allow the server to call other servers while impersonating the original client.

All other fields are as specified in [MS-CIFS] section 2.2.4.64.1.<51>

```
SMB_Parameters
   {
  UCHAR WordCount;
  Words
     {
     UCHAR          AndXCommand;
     UCHAR          AndXReserved;
     USHORT         AndXOffset;
     UCHAR          Reserved;
     USHORT         NameLength;
     ULONG          Flags;
     ULONG          RootDirectoryFID;
     ULONG          DesiredAccess;
     LARGE_INTEGER  AllocationSize;
     ULONG          ExtFileAttributes;
     ULONG          ShareAccess;
     ULONG          CreateDisposition;
     ULONG          CreateOptions;
     ULONG          ImpersonationLevel;
     UCHAR          SecurityFlags;
     }
   }
  SMB_Data
   {
  USHORT ByteCount;
  Bytes
     {
     SMB_STRING FileName;
     }
   }
```

SMB_Parameters

Words (48 bytes):

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
Reserved								NameLength																Flags							
...																								RootDirectoryFID							
...																								DesiredAccess							

Release: Tuesday, April 6, 2010

...		AllocationSize
...		
...		ExtFileAttributes
...		ShareAccess
...		CreateDisposition
...		CreateOptions
...		ImpersonationLevel
...		SecurityFlags

Flags (4 bytes): A set of flags that modify the client request, as defined in the table below. NT_CREATE_REQUEST_EXTENDED_RESPONSE is new to MS-SMB. All other flags are included in the table for completeness. Unused bits SHOULD be set to zero by the client when sending a request and MUST be ignored when received by the server.

Name & bitmask	Meaning
NT_CREATE_REQUEST_OPLOCK 0x00000002	If set, then the client is requesting an oplock.
NT_CREATE_REQUEST_OPBATCH 0x00000004	If set, then the client is requesting a batch oplock.
NT_CREATE_OPEN_TARGET_DIR 0x00000008	If set, then the client indicates that the parent directory of the target is to be opened. If the target does exist, then it might require a delete access check for the parent directory.
NT_CREATE_REQUEST_EXTENDED_RESPONSE 0x00000010	If set, then the client is requesting extended information in the response.

ImpersonationLevel (4 bytes): This field specifies the information given to the server about the client and how the server SHOULD<52> represent or impersonate the client. Security impersonation levels govern the degree to which a server process can act on behalf of a client process. This field MUST be set to one of the following values. Impersonation is described in [MS-WSO] section 3.1.2.4; for more information about impersonation, see [MSDN-IMPERS].

Value	Meaning
SECURITY_ANONYMOUS 0x00000000	The server cannot obtain identification information about the client, and it cannot impersonate the client.
SECURITY_IDENTIFICATION	The server can obtain security information about the client, such as

Value	Meaning
0x00000001	**security identifiers (SIDs)** and privileges, but it cannot impersonate the client. This is useful for servers that export their own objects; for example, database products that export tables and views. Using the retrieved client-security information, the server can make access-validation decisions without being capable of using other services that are using the client's **security context**.
SECURITY_IMPERSONATION 0x00000002	The server can impersonate the client's security context on the server system. The server cannot impersonate the client on other remote systems.
SECURITY_DELEGATION 0x00000003	The server can impersonate the client's security context on the server system and on remote systems.

2.2.4.9.2 Server Response Extensions

A successful response takes the following format. If the server receives more than one SMB_COM_NT_CREATE_ANDX request from a client before it sends back any response, then the server can respond to these requests in any order.

When a client requests extended information, then the response takes the form described below. Aside from the **WordCount**, **NMPipeStatus_or_FileStatusFlags**, **FileId**, **VolumeGUID**, **FileId**, **MaximalAccessRights**, and **GuestMaximalAccessRights** fields, all other fields are as specified in [MS-CIFS] section 2.2.4.64.2.

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR          AndXCommand;
    UCHAR          AndXReserved;
    USHORT         AndXOffset;
    UCHAR          OplockLevel;
    USHORT         FID;
    ULONG          CreationAction;
    FILETIME       CreateTime;
    FILETIME       LastAccessTime;
    FILETIME       LastWriteTime;
    FILETIME       LastChangeTime;
    ULONG          ExtFileAttributes;
    LARGE_INTERGER AllocationSize;
    LARGE_INTERGER EndOfFile;
    USHORT         ResourceType;
    USHORT         NMPipeStatus_or_FileStatusFlags;
    UCHAR          Directory;
    GUID           VolumeGUID;
    ULONGLONG      FileId;
    ACCESS_MASK    MaximalAccessRights;
    ACCESS_MASK    GuestMaximalAccessRights;
    }
  }
  SMB_Data
    {
    USHORT ByteCount;
```

```
    }
```

SMB_Parameters:

WordCount (1 bytes): This field SHOULD<53> be 0x32.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
AndXCommand								AndXReserved								AndXOffset															
OplockLevel						FID														CreationAction											
...																								CreateTime							
...																															
...																								LastAccessTime							
...																															
...																								LastWriteTime							
...																															
...																								LastChangeTime							
...																															
...																								ExtFileAttributes							
...																								AllocationSize							
...																															
...																								EndOfFile							
...																															
...																								ResourceType							
...						NMPipeStatus_or_FileStatusFlags																Directory							

VolumeGUID
...
...
...
FileId
...
MaximalAccessRights
GuestMaximalAccessRights

ResourceType (2 bytes): The file type. This field SHOULD<54> be interpreted as follows:

Name & value	Meaning
FileTypeDisk 0x0000	File or Directory
FileTypeByteModePipe 0x0001	Byte mode named pipe
FileTypeMessageModePipe 0x0002	Message mode named pipe
FileTypePrinter 0x0003	Printer Device
FileTypeUnknown 0xFFFF	Unknown file type

NMPipeStatus_or_FileStatusFlags (2 bytes): A union between the **NMPipeStatus** field and the new **FileStatusFlags** field. If the **ResourceType** field is a named pipe (**FileTypeByteModePipe** or **FileTypeMessageModePipe**), then this field MUST be the **NMPipeStatus** field:

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
NMPipeStatus																FileStatusFlags															

NMPipeStatus (2 bytes): A 16-bit field that shows the status of the opened named pipe. This field is formatted as an SMB_NMPIPE_STATUS ([MS-CIFS] section 2.2.1.3).

If the **ResourceType** field is FileTypeDisk, then this field MUST be the **FileStatusFlags** field:

FileStatusFlags (2 bytes): A 16-bit field that shows extra information about the opened file or directory. Any combination of the following flags is valid. Unused bit fields SHOULD be set to zero by the server and MUST be ignored by the client.

Name & bitmask	Meaning
NO_EAS 0x0001	The file or directory has no extended attributes.
NO_SUBSTREAMS 0x0002	The file or directory has no data streams other than the main data stream.
NO_REPARSETAG 0x0004	The file or directory is not a reparse point.

For all other values of **ResourceType**, this field SHOULD be set to zero by the server when sending a response and MUST be ignored when received by the client.

VolumeGUID (16 bytes): This field MUST be a GUID value that uniquely identifies the volume on which the file resides. This field MUST zero if the underlying file system does not support volume GUIDs.<55>

FileId (8 bytes): This field MUST be a 64-bit opaque value that uniquely identifies this file on a volume. This field MUST be set to zero if the underlying file system does not support unique FileId numbers on a volume. If the underlying file system does support unique FileId numbers, then this value SHOULD<56> be set to the unique FileId for this file.

MaximalAccessRights (4 bytes): The maximum access rights that the user opening the file has been granted for this file open. This field MUST be encoded in an ACCESS_MASK format, as specified in section 2.2.1.4.

GuestMaximalAccessRights (4 bytes): The maximum access rights that the guest account has when opening this file. This field MUST be encoded in an ACCESS_MASK format, as specified in section 2.2.1.4. Note that the notion of a guest account is implementation specific. Implementations that do not support the notion of a guest account MUST set this field to zero.<57>

SMB_Data:

ByteCount (2 bytes): This field SHOULD<58> be zero.

2.2.5   Transaction Subcommand Extensions

2.2.5.1   TRANS_RAW_READ_NMPIPE (0x0036)

The TRANS_RAW_READ_NMPIPE subcommand allows for a **raw read** of data from a named pipe. This method of reading data from a named pipe ignores message boundaries even if the pipe was set up as a message mode pipe.

The status of this subcommand is **obsolescent**.<59> Aside from its new status as obsolescent, this subcommand is exactly as described in [MS-CIFS] section 2.2.5.2.

2.2.5.2   TRANS_CALL_NMPIPE (0x0054)

The TRANS_CALL_NMPIPE subcommand allows a client to connect to a named pipe, write to the named pipe, read from the named pipe, and close the named pipe in a single command.

The status of this subcommand is obsolescent.<60> Aside from its new status as obsolescent, this subcommand is exactly as described in [MS-CIFS] section 2.2.5.11.

2.2.6   Transaction 2 Subcommand Extensions

2.2.6.1   TRANS2_FIND_FIRST2 (0x0001)

2.2.6.1.1   Client Request Extensions

A TRANS2_FIND_FIRST2 subcommand of SMB_COM_TRANSACTION2 is sent by the client to retrieve an enumeration of files, as specified in [MS-CIFS] section 2.2.6.2.1.

The list of valid Information Level values has been extended, as specified in section 2.2.2.3.1, to include SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO and SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO. Extensions are also presented to the SMB_FIND_FILE_BOTH_DIRECTORY_INFO Information Level, as specified in section 2.2.8.1.1. This Information Level now provides support for accessing enumerations of available previous version timestamps of files or directories.

Aside from the Information Level extensions and the **FileName** field, all of the other fields are as specified in [MS-CIFS] 2.2.6.2.1.

Trans2_Parameters

FileName (variable): This field is extended to support use of the @GMT token wildcard (section 2.2.1.1.1).<61> If this character sequence contains the @GMT-* wildcard, **Trans2_Data.InformationLevel** SHOULD be set to SMB_FIND_FILE_BOTH_DIRECTORY_INFO.<62>

Trans2_Data

InformationLevel (2 bytes): This field contains an Information Level code, which determines the information contained in the response. The list of valid Information Level codes is specified in section 2.2.2.3.1. A client that has not negotiated long names support MUST request only SMB_INFO_STANDARD. If a client that has not negotiated long names support requests an **InformationLevel** other than SMB_INFO_STANDARD, then the server MUST return a status of STATUS_INVALID_PARAMETER (ERRDOS/ERRinvalidparam).

2.2.6.1.2   Server Response Extensions

A server MUST send a TRANS2_FIND_FIRST2 response in reply to a client TRANS2_FIND_FIRST2 subcommand request when the request is successful, as specified in [MS-CIFS] section 2.2.6.2.2.

Thus subcommand also supports two new Information Levels, as well as extensions to the SMB_FIND_FILE_BOTH_DIRECTORY_INFO Information Level, as defined in section 2.2.2.3.1. The format of the file information returned for these two Information Levels (and the format of information for a TRANS2_FIND_FIRST2 response for the special FileName pattern @GMT token) is listed in section 2.2.8.1.If a client does not support long names (that is, SMB_FLAGS2_KNOWS_LONG_NAMES is not set in the **Flags2** field of the SMB Header), then any TRANS2_FIND_FIRST2 request with an Information Level other than SMB_INFO_STANDARD, or any

TRANS2_FIND_NEXT2 request with an Information Level other than SMB_INFO_STANDARD or SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO, MUST be failed with STATUS_INVALID_PARAMETER.

2.2.6.2   TRANS2_FIND_NEXT2 (0x0002)

2.2.6.2.1   Client Request Extensions

The TRANS2_FIND_NEXT2 subcommand of the SMB_COM_TRANSACTION2 request is sent by a client to continue a file enumeration, as specified in [MS-CIFS] sections 2.2.6.3.1. The request format is identical to the request format that is specified in [MS-CIFS] section 2.2.6.3.1, except that two new Information Levels have been added and one has been extended. See section 2.2.6.1.1 for details.

2.2.6.2.2   Server Response Extensions

The server MUST send a TRANS2_FIND_NEXT2 response in reply to a client TRANS2_FIND_NEXT2 subcommand request when the request is successful. The format of this packet is identical to what is specified in [MS-CIFS] section 2.2.6.3.2, except that two new Information Levels have been added and one has been extended. See section 2.2.6.1.2 for details.

2.2.6.3   TRANS2_QUERY_FS_INFORMATION (0x0003)

This subcommand supports new pass-through Information Level capabilities, as specified in section 2.2.2.3.5.

2.2.6.3.1   Client Request Extensions

A TRANS2_QUERY_FS_INFORMATION subcommand of the SMB_COM_TRANSACTION2 is sent by the client to request attribute information about the file system, as specified in [MS-CIFS] section 2.2.6.4.1.

2.2.6.3.2   Server Response Extensions

A server MUST send a TRANS2_QUERY_FS_INFORMATION response in reply to an SMB_COM_TRANSACTION2 client request with a TRANS2_QUERY_FS_INFORMATION subcommand when the request is successful, as specified in [MS-CIFS] section 2.2.6.4.2.

2.2.6.4   TRANS2_SET_FS_INFORMATION (0x0004)

The TRANS2_SET_FS_INFORMATION subcommand of the SMB_COM_TRANSACTION2 command (section 2.2.4.4) is sent by the client to set file system attribute information on the server. This subcommand was introduced in the LAN Manager 2.0 dialect.<63>

The TRANS2_SET_FS_INFORMATION subcommand of the SMB_COM_TRANSACTION2 command (section 2.2.4.4) is sent by the client to set file system attribute information on the server.

The TRANS2_SET_FS_INFORMATION request and response formats are special cases of SMB_COM_TRANSACTION2 command. Only the TRANS2_SET_FS_INFORMATION specifics are described here.

2.2.6.4.1   Client Request

  SMB_Parameters:

  WordCount (1 byte): This field MUST be 0x0F.

Words (30 bytes):

TotalParameterCount (2 bytes): This field MUST be 0x0004.

TotalDataCount (2 bytes): This field MUST be greater than or equal to 0x0000.

SetupCount (1 byte): This field MUST be 0x01.

Setup (2 bytes): This field MUST be TRANS2_SET_FS_INFORMATION (0x0004).

Trans2_Parameters

```
Trans2_Parameters
  {
  USHORT FID;
  USHORT InformationLevel;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
FID																InformationLevel															

FID (2 bytes): A valid Fid that represents the open file on the server that is to have its file attributes changed.

InformationLevel (2 bytes): The Information Level of the request. This field MUST be a pass-through Information Level, as described in section 2.2.2.3.5.

Trans2_Data

The Trans2_Data block carries the structure of the Information Level specified by the request's **Trans2_Parameters.InformationLevel** field. Since this subcommand only accepts pass-through Information Levels, the structure of this section is implementation specific.

2.2.6.4.2 Server Response

A server MUST send a TRANS2_SET_FS_INFORMATION response in reply to a client TRANS2_SET_FS_INFORMATION subcommand request when the request is successful. The server MUST set an error code in the Status field of the SMB header of the response to indicate whether the request was successful or failed.<64>

Trans2_Parameters

No Trans2_Parameters are sent by this message.

Trans2_Data

No Trans2_Data is sent by this message.

Error Codes

SMB error class	SMB error code	NT status code	POSIX equivalent	Description
ERRDOS(0x01)	ERRnoaccess(0x0005)	STATUS_ACCESS_DENIED(0xC0000022)	EPERM	Access denied.
	ERRbadfid(0x0006)	STATUS_INVALID_HANDLE(0xC0000008)	ENOENT	The Fid supplied is invalid.
	ERRnomem(0x0008)	STATUS_INSUFF_SERVER_RESOURCES(0xC0000205)	ENOMEM	The server is out of resources.
	ERRunsup(0x0032)	STATUS_NOT_SUPPORTED(0xC00000BB)		The supplied Information Level is not a valid pass-through Information Level.
	ERRinvalidparam(0x0057)	STATUS_INVALID_PARAMETER(0xC000000D)		The supplied pass-through Information Level values contain at least one invalid parameter.
ERRSRV(0x02)	ERRerror(0x0001)	STATUS_INVALID_SMB(0x00010002)		Invalid SMB. Not enough parameter bytes were sent.
	ERRinvtid(0x0005)	STATUS_INVALID_HANDLE(0xC0000008)STATUS_SMB_BAD_TID(0x00050002)		The TID is no longer

SMB error class	SMB error code	NT status code	POSIX equivalent	Description
				valid.
	ERRbaduid(0x005B)	STATUS_INVALID_HANDLE(0xC0000008)STATUS_SMB_BAD_UID(0x005B0002)		The UID supplied is not known to the session.
ERRHRD(0x03)	ERRnowrite(0x0013)	STATUS_MEDIA_WRITE_PROTECTED(0xC00000A2)		The Fid supplied is on write-protected media.
	ERRdata(0x0017)	STATUS_DATA_ERROR(0xC000003E)	EIO	Disk I/O error.

2.2.6.5   TRANS2_QUERY_PATH_INFORMATION (0x0005)

This subcommand supports new pass-through Information Level capabilities, as specified in section 2.2.2.3.5.

2.2.6.5.1   Client Request Extensions

A TRANS2_QUERY_PATH_INFORMATION subcommand of SMB_COM_TRANSACTION2 is sent by the client to request attribute information for a file or directory by specifying the path, as specified in [MS-CIFS], section 2.2.6.6.1.

2.2.6.5.2   Server Response Extensions

A server MUST send a TRANS2_QUERY_PATH_INFORMATION response in reply to an SMB_COM_TRANSACTION2 client request with a TRANS2_QUERY_PATH_INFORMATION subcommand when the request is successful, as specified in [MS-CIFS] section 2.2.6.6.2.

2.2.6.6   TRANS2_SET_PATH_INFORMATION (0x0006)

This subcommand supports new pass-through Information Level capabilities, as specified in section 2.2.2.3.5.

2.2.6.6.1   Client Request Extensions

A TRANS2_SET_PATH_INFORMATION subcommand of SMB_COM_TRANSACTION2 is sent by the client to request a change of attribute information for a file or directory by path, as specified in [MS-CIFS] section 2.2.6.7.1.

2.2.6.6.2   Server Response Extensions

A server MUST send a TRANS2_SET_PATH_INFORMATION response in reply to an SMB_COM_TRANSACTION2 (section 2.2.4.4) client request with a

TRANS2_SET_PATH_INFORMATION subcommand when the request is successful,as specified in [MS-CIFS] section 2.2.6.7.2.

2.2.6.7   TRANS2_QUERY_FILE_INFORMATION (0x0007)

This subcommand supports new pass-through Information Level capabilities, as specified in section 2.2.2.3.5.

2.2.6.7.1   Client Request Extensions

A TRANS2_QUERY_FILE_INFORMATION subcommand of SMB_COM_TRANSACTION2 is sent by a client to request attribute information for a file or directory that has been opened, as specified in [MS-CIFS] section 2.2.6.8.1.

2.2.6.7.2   Server Response Extensions

A server MUST send a TRANS2_QUERY_PATH_INFORMATION response in reply to an SMB_COM_TRANSACTION2 client request with a TRANS2_QUERY_PATH_INFORMATION subcommand when the request is successful. The Trans2_Data block of the transaction response contains the information requested by the Information Level in the request, as specified in [MS-CIFS] section 2.2.6.8.2.

2.2.6.8   TRANS2_SET_FILE_INFORMATION (0x0008)

This subcommand supports new pass-through Information Level capabilities, as specified in section 2.2.2.3.5.

2.2.6.8.1   Client Request Extensions

A TRANS2_SET_FILE_INFORMATION subcommand of SMB_COM_TRANSACTION2 is sent by the client to request a change of attribute information for a file or directory that has been opened, as specified in [MS-CIFS] section 2.2.6.9.1.

2.2.6.8.2   Server Response Extensions

A server MUST send a TRANS2_SET_FILE_INFORMATION response in reply to an SMB_COM_TRANSACTION2 client request with a TRANS2_SET_FILE_INFORMATION subcommand when the request is successful, as specified in [MS-CIFS] section 2.2.6.9.2.

2.2.7   NT Transact Subcommand Extensions

2.2.7.1   NT_TRANSACT_CREATE (0x0001) Extensions

An SMB_COM_NT_CREATE_ANDX request is sent by a client to open a file or device on the server. This extension adds an additional parameter value to the **ImpersonationLevel** field. SECURITY_DELEGATION is added to allow the server to call other servers while impersonating the original client.

ImpersonationLevel (4 bytes): This field specifies the information given to the server about the client and how the server SHOULD<65> represent or impersonate the client. Security impersonation levels govern the degree to which a server process can act on behalf of a client process. This field MUST be set to one of the following values. Impersonation is described in [MS-WSO] section 3.1.2.4; for more information about impersonation, see [MSDN-IMPERS].

Value	Meaning
SECURITY_ANONYMOUS 0x00000000	The server cannot obtain identification information about the client, and it cannot impersonate the client.
SECURITY_IDENTIFICATION 0x00000001	The server can obtain security information about the client, such as security identifiers (SIDs) and privileges, but it cannot impersonate the client. This is useful for servers that export their own objects; for example, database products that export tables and views. Using the retrieved client-security information, the server can make access-validation decisions without being capable of using other services that are using the client's security context.
SECURITY_IMPERSONATION 0x00000002	The server can impersonate the client's security context on the server system. The server cannot impersonate the client on other remote systems.
SECURITY_DELEGATION 0x00000003	The server can impersonate the client's security context on the server system and on remote systems.

2.2.7.2 NT_TRANSACT_IOCTL (0x0002)

An SMB_COM_NT_TRANSACT (section 2.2.4.8) command with an NT_TRANSACT_IOCTL subcommand is sent by a client to pass an IOCTL or file system control (FSCTL) command to a server. The NT_TRANSACT_IOCTL subcommand is specified in [MS-CIFS] section 2.2.7.2.

2.2.7.2.1 Client Request Extensions

The NT_TRANSACT_IOCTL request is a special case of the SMB_COM_NT_TRANSACT command request. Only the NT_TRANSACT_IOCTL specifics are described here.

The FSCTL operations listed in the table below are new to these extensions and only those are specific to the SMB protocol.<66> Other FSCTL and IOCTL control codes are not defined in this specification and are specific to the underlying object store of the server.<67> If an application requests an undefined FSCTL or IOCTL operation, then the client SHOULD still pass the request through to the server. Server implementations that receive undefined FSCTL or IOCTL operation requests MUST either pass the operation code and data (if any) through to the underlying object store or fail the operation by returning STATUS_NOT_SUPPORTED.<68>

Name	Value	Description
FSCTL_SRV_ENUMERATE_SNAPSHOTS	0x00144064	Enumerate previous versions of a file.
FSCTL_SRV_REQUEST_RESUME_KEY	0x00140078	Retrieve an opaque file reference for server-side data movement.<69>
FSCTL_SRV_COPYCHUNK	0x001440F2	Perform server-side data movement.<70>

FSCTL_SRV_ENUMERATE_SNAPSHOTS Request

This FSCTL is used to enumerate available previous version timestamps (snapshots) of a file or directory.

The FSCTL_SRV_ENUMERATE_SNAPSHOTS request format is a special case of the NT_TRANSACT_IOCTL subcommand. Only the FSCTL_SRV_ENUMERATE_SNAPSHOTS request specifics are described here.

SMB_Parameters

    WordCount (1 byte): The value of this field MUST be 0x17.

    Words (46 bytes):

        MaxDataCount (4 bytes): This field MUST be greater than or equal to 0x000D.<71>

        Setup (8 bytes):

        FunctionCode (4 bytes): This field MUST be 0x00144064.

        FID (2 bytes): This field MUST contain a valid Fid representing a valid Open on a file. This is the file for which snapshots are being requested.

        IsFctl (1 byte): MUST be TRUE (any non-zero value).

        IsFlags (1 byte) : MUST be zero (0x00).

NT_Trans_Parameters

No NT Trans parameters are sent in this request.

NT_Trans_Data

No NT Trans data is sent in this request.

FSCTL_SRV_REQUEST_RESUME_KEY Request

This FSCTL is used to retrieve an opaque file reference for server-side data movement operations, as specified in section 3.2.4.11.2.

The FSCTL_SRV_REQUEST_RESUME_KEY request format is a special case of the NT_TRANSACT_IOCTL subcommand. Only the FSCTL_SRV_REQUEST_RESUME_KEY request specifics are described here.

SMB_Parameters

    WordCount (1 byte): The value of this field MUST be 0x17.

    Words (46 bytes):

        MaxDataCount (4 bytes): This field MUST be greater than or equal to 0x001D.

        Setup (8 bytes):

        FunctionCode (4 bytes): This field MUST be 0x00140078.

        FID (2 bytes): This field MUST contain a valid Fid that represents a valid Open on a file. This file is the source file for a server-side data copy operation.

        IsFctl (1 byte): MUST be TRUE (any non-zero value).

        IsFlags (1 byte): MUST be zero (0x00).

NT_Trans_Parameters

No NT Trans parameters are sent in this request.

NT_Trans_Data

No NT Trans data is sent in this request.

FSCTL_SRV_COPYCHUNK Request

This FSCTL is used for server-side data movement, as specified in section 3.2.4.11.2.

The FSCTL_SRV_COPYCHUNK request format is a special case of NT_TRANSACT_IOCTL subcommand. Only the FSCTL_SRV_COPYCHUNK request specifics are described here.

SMB_Parameters

   WordCount (1 byte): The value of this field MUST be 0x17.

   Words (46 bytes):

      TotalDataCount (4 bytes): This field MUST be greater than or equal to 0x0034.

      MaxDataCount (4 bytes): This field MUST be greater than or equal to 0x001D.

      Setup (8 bytes):

      FunctionCode (4 bytes): This field MUST be 0x00144078.

      FID (2 bytes): This field MUST contain a valid Fid that represents a valid Open on a file. This file is the destination file for a server-side data copy operation.

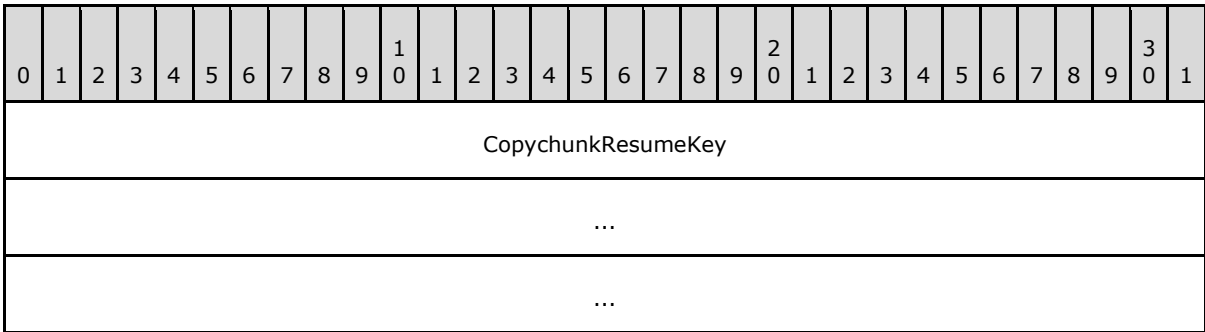   **IsFctl (1 byte):** This field MUST be TRUE (any non-zero value).

      IsFlags (1 byte): The value of this field MUST be zero (0x00).

NT_Trans_Parameters

No NT Trans parameters are sent in this request.

NT_Trans_Data

```
NT_Trans_Data
  {
  COPYCHUNK_RESUME_KEY  CopychunkResumeKey;
  ULONG                 ChunkCount;
  ULONG                 Reserved;
  SRV_COPYCHUNK         CopychunkList[ChunkCount];
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
CopychunkResumeKey																															
...																															
...																															

...	
...	
...	
ChunkCount	
Reserved	
CopychunkList (variable)	
...	

CopychunkResumeKey (24 bytes): An opaque 24-byte server copychunk resume key for a source file in a server-side file copy operation. This field value is received from a previous FSCTL_SRV_REQUEST_RESUME_KEY server response.

ChunkCount (4 bytes): The number of entries, or "copychunks", in the **CopyChunkList**. This field also represents the number of server-side data movement operations being requested. This field MUST NOT be zero.

Reserved (4 bytes): A reserved field. This field SHOULD be set to zero when sending the request. This field MUST be ignored by the server when the message is received.

CopychunkList (variable): A concatenated list of copychunks. Each entry is formatted as a SRV_COPYCHUNK structure.

2.2.7.2.1.1 SRV_COPYCHUNK

```
SRV_COPYCHUNK
  {
  LARGE_INTEGER SourceOffset;
  LARGE_INTEGER DestinationOffset;
  ULONG         CopyLength;
  ULONG         Reserved;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SourceOffset																															
...																															
DestinationOffset																															

...
CopyLength
Reserved

SourceOffset (8 bytes): The offset, in bytes, into the source file from which data is being copied.

DestinationOffset (8 bytes): The offset, in bytes, into the destination file to which data is being copied.

CopyLength (4 bytes): The number of bytes to copy from the source file to the destination file.

Reserved (4 bytes): This field SHOULD<72> be set to zero by the client and MUST be ignored upon receipt by the server.

2.2.7.2.2 Server Response Extensions

An SMB_COM_NT_TRANSACT (section 2.2.4.8) response for an NT_TRANSACT_IOCTL ([MS-CIFS] section 2.2.7.2) subcommand MUST be sent by a server in reply to a successful NT_TRANSACT_IOCTL request.

The NT_TRANSACT_IOCTL response is a special case of the SMB_COM_NT_TRANSACT command response. Only the NT_TRANSACT_IOCTL specifics are described here.

2.2.7.2.2.1 FSCTL_SRV_ENUMERATE_SNAPSHOTS Response

The FSCTL_SRV_ENUMERATE_SNAPSHOTS response format is a special case of the NT_TRANSACT_IOCTL subcommand. Only the FSCTL_SRV_ENUMERATE_SNAPSHOTS response specifics are described here.

NT_Trans_Parameters

No NT Trans parameters are sent in this response.

NT_Trans_Data

```
NT_Trans_Data
   {
   ULONG NumberOfSnapShots;
   ULONG NumberOfSnapShotsReturned;
   ULONG SnapShotArraySize;
   WCHAR SnapShotMultiSZ[];
   }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
NumberOfSnapShots																																

NumberOfSnapShotsReturned
SnapShotArraySize
SnapShotMultiSZ (variable)
...

NumberOfSnapShots (4 bytes): The number of snapshots that the underlying object store contains of this file.

NumberOfSnapShotsReturned (4 bytes): This value MUST be the number of snapshots that are returned in this response. If this value is less than **NumberofSnapshots**, then there are more snapshots than were able to fit in this response.

SnapShotArraySize (4 bytes): The length, in bytes, of the **SnapShotMultiSZ** field.

SnapShotMultiSZ (variable): A concatenated list of available snapshots. Each snapshot MUST be encoded as a NULL-terminated sequence of 16-bit Unicode characters, and MUST take on the following form: @GMT-YYYY.MM.DD-HH.MM.SS. The concatenated list MUST be terminated by two additional 16-bit Unicode NULL characters. If the response contains no snapshots, then the server SHOULD set this field to two 16-bit Unicode NULL characters.

2.2.7.2.2.2   FSCTL_SRV_REQUEST_RESUME_KEY Response

The FSCTL_SRV_REQUEST_RESUME_KEY response format is a special case of the NT_TRANSACT_IOCTL subcommand. Only the FSCTL_SRV_REQUEST_RESUME_KEY response specifics are described here.

Although this FSCTL includes support for returning extended context information for a copychunk resume key, this feature is considered reserved but not implemented and is not used in this response.

NT_Trans_Parameters

No NT Trans parameters are sent in this response.

NT_Trans_Data

```
NT_Trans_Data
   {
   COPYCHUNK_RESUME_KEY CopychunkResumeKey;
   ULONG                ContextLength;
   UCHAR                Context[ContextLength] (optional);
   }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
															CopychunkResumeKey																	

...
...
...
...
...
ContextLength
Context (variable)
...

CopychunkResumeKey (24 bytes): A 24-byte copychunk resume key generated by the server that can be subsequently used by the client to uniquely identify the open source file in a FSCTL_SRV_COPYCHUNK request. The client MUST NOT attach any interpretation to this key and MUST treat it as an opaque value. For more information, see section 3.3.5.10.1.

ContextLength (4 bytes): The length, in bytes, of the **Context** field. Since this feature is not used, this field SHOULD be set to zero by the server and MUST be ignored by the client.

Context (variable): The copychunk resume key's extended context information. Since this feature is not used, this field SHOULD NOT<73> be included in the response.

2.2.7.2.2.3   FSCTL_SRV_COPYCHUNK Response

The FSCTL_SRV_COPYCHUNK response format is a special case of NT_TRANSACT_IOCTL subcommand. Only the FSCTL_SRV_COPYCHUNK response specifics are described here.

NT_Trans_Parameters

No NT Trans parameters are sent in this response.

NT_Trans_Data

```
NT_Trans_Data
  {
  ULONG ChunksWritten;
  ULONG ChunkBytesWritten;
  ULONG TotalBytesWritten;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ChunksWritten																															
ChunkBytesWritten																															
TotalBytesWritten																															

ChunksWritten (4 bytes): This field MUST represent the number of copychunk operations successfully processed by the server.

ChunkBytesWritten (4 bytes): This field is unused. This field MUST be set to zero by the server and MUST be ignored by the client.

TotalBytesWritten (4 bytes): This field MUST represent the total number of bytes written to the destination file across all copychunk operations.

2.2.7.3   NT_TRANSACT_QUERY_QUOTA (0x0007)

An SMB_COM_NT_TRANSACT (section 2.2.4.8) command with an NT_TRANSACT_QUERY_QUOTA subcommand code is used by a client to query quota information for a user or multiple users. This subcommand is new to these extensions.

2.2.7.3.1   Client Request

The NT_TRANSACT_QUERY_QUOTA request is a special case of the SMB_COM_NT_TRANSACT command request. Only the NT_TRANSACT_QUERY_QUOTA specifics are described here.

At least one of **NT_Trans_Parameters.SidListLength** or **NT_Trans_Parameters.StartSidLength** MUST be zero. If both are zero, then the quota information for all SIDs on the underlying object store of a share MUST be enumerated by the server.

SMB_Parameters

WordCount (1 byte): The value of this field MUST be 0x13.

Words (38 bytes):

SetupCount (1 bytes): This field MUST be 0x00.

NT_Trans_Parameters:

```
NT_Trans_Parameters
  {
  USHORT   FID;
  BOOLEAN  ReturnSingleEntry;
  BOOLEAN  RestartScan;
  ULONG    SidListLength;
  ULONG    StartSidLength;
  ULONG    StartSidOffset;
```

```
      }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
FID																ReturnSingleEntry								RestartScan							
SidListLength																															
StartSidLength																															
StartSidOffset																															

FID (2 bytes): An Fid to a file or directory. The quota information of the object store underlying the file or directory MUST be queried.

ReturnSingleEntry (1 byte): If TRUE (any non-zero value), then this field indicates that the server behavior is to be restricted to only return a single SID's quota information instead of filling the entire buffer.

RestartScan (1 byte): If TRUE (any non-zero value), then this field indicates that the scan of quota information is to be restarted.

SidListLength (4 bytes): If non-zero, then this field indicates that the client is requesting quota information of a particular set of SIDs and MUST represent the length of the **NT_Trans_Data.SidList** field.

StartSidLength (4 bytes): If non-zero, then this field indicates that the **SidList** field contains a start SID (that is, a single SID entry indicating to the server where to start user quota information enumeration) and MUST represent the length, in bytes, of that **SidList** entry.

StartSidOffset (4 bytes): If **StartSidLength** is non-zero, then this field MUST represent the offset from the start of the NT_Trans_Data to the specific **SidList** entry at which to begin user quota information enumeration. Otherwise, this field SHOULD be set to zero and MUST be ignored by the server.

NT_Trans_Data :

```
NT_Trans_Data
  {
  FILE_GET_QUOTA_INFORMATION SidList[] (optional);
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SidList (variable)																															

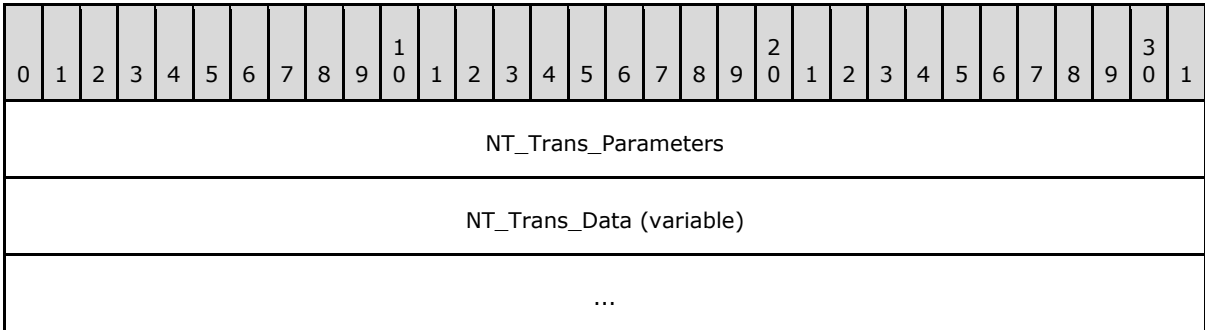|																														
															...															

SidList (variable): A list of one or more SIDs that are formatted as a FILE_GET_QUOTA_INFORMATION structure, as specified in [MS-FSCC] section 2.4.33.1. If **SidListLength** is non-zero, then this field MUST contain a list of one or more SIDs for which quota information is being requested. If **StartSidLength** is non-zero, then this field MUST contain a start SID. If both **SidListLength** and **StartSidLength** are zero, then this field MUST NOT be included in the request.

2.2.7.3.2 Server Response

An SMB_COM_NT_TRANSACT (section 2.2.4.8) response for an NT_TRANSACT_QUERY_QUOTA subcommand MUST be sent by a server in reply to a client NT_TRANSACT_QUERY_QUOTA subcommand request when the request is successful.
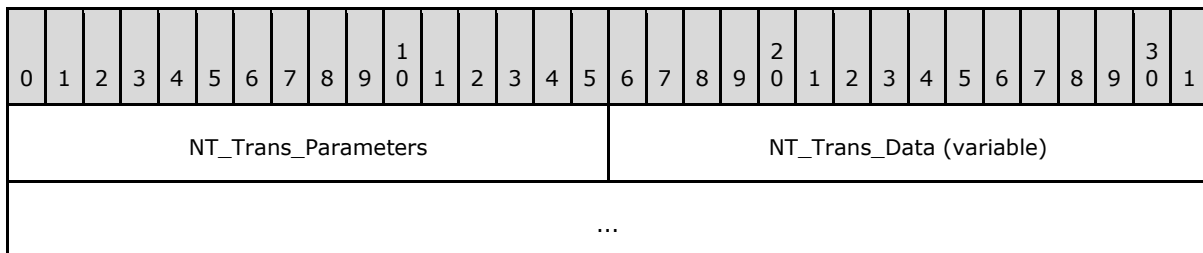
The NT_TRANSACT_QUERY_QUOTA response is a special case of the SMB_COM_NT_TRANSACT command response. Only the NT_TRANSACT_QUERY_QUOTA specifics are described here.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
										NT_Trans_Parameters																					
										NT_Trans_Data (variable)																					
										...																					

NT_Trans_Parameters (4 bytes):

```
NT_Trans_Parameters
  {
  ULONG DataLength;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
										DataLength																					
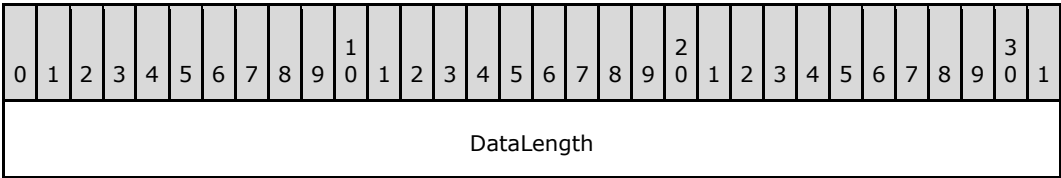
DataLength (4 bytes): The length, in bytes, of the returned user quota information. This field MUST be equal to the **SMB_Parameters.Words.TotalDataCount** field.

NT_Trans_Data (variable):
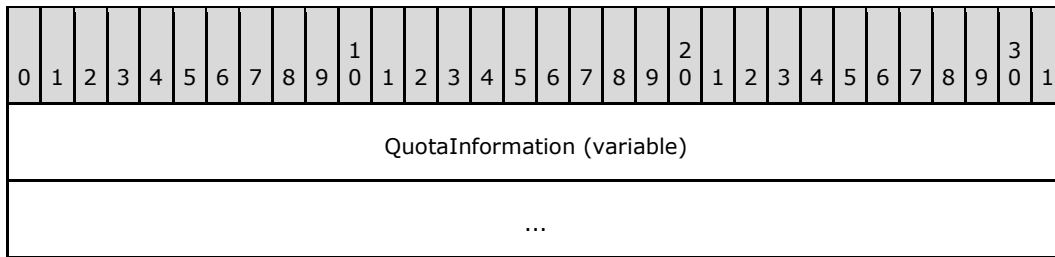
```
NT_Trans_Data
  {
  FILE_QUOTA_INFORMATION QuotaInformation[] (variable);
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
											QuotaInformation (variable)																				
											...																				

QuotaInformation (variable): A concatenated list of FILE_QUOTA_INFORMATION structures, as specified in [MS-FSCC] section 2.4.33.

2.2.7.4   NT_TRANSACT_SET_QUOTA (0x0008)

An SMB_COM_NT_TRANSACT (section 2.2.4.8) request with an NT_TRANSACT_SET_QUOTA subcommand code is sent by a client to set user quota information on the underlying object store of a server. This subcommand is new to these extensions.

2.2.7.4.1   Client Request

The NT_TRANSACT_SET_QUOTA request is a special case of the SMB_COM_NT_TRANSACT command request. Only the NT_TRANSACT_SET_QUOTA specifics are described here.

   SMB_Parameters:

   WordCount (1 byte): The value of this field MUST be 0x13.

   Words (38 bytes):

   SetupCount (1 bytes): This field MUST be 0x00.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
						NT_Trans_Parameters														NT_Trans_Data (variable)											
											...																				

   NT_Trans_Parameters (2 bytes):

```
NT_Trans_Parameters
  {
  USHORT FID;
  }
```
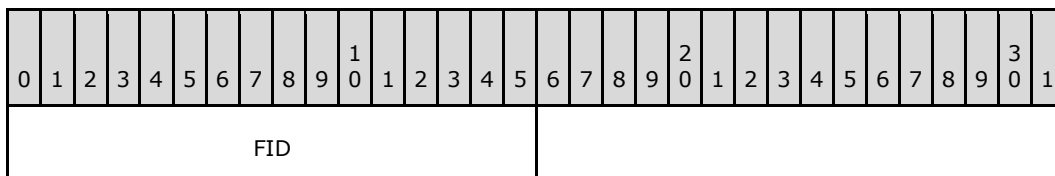
0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
						FID																									

FID (2 bytes): An Fid to a file or directory. The quota information of the object store underlying the file or directory MUST be modified.

NT_Trans_Data (variable):
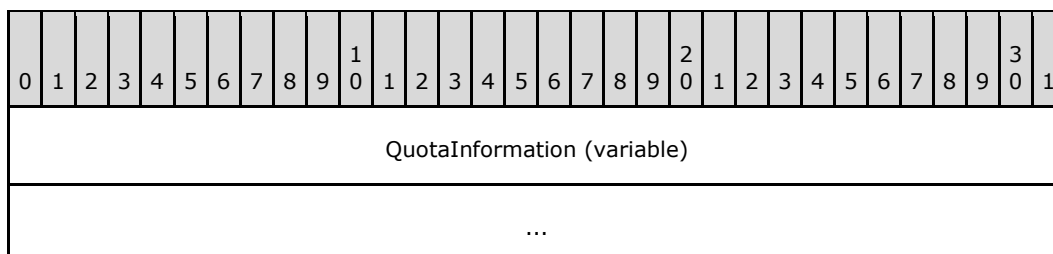
```
NT_Trans_Data
  {
  FILE_QUOTA_INFORMATION QuotaInformation[] (variable);
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
QuotaInformation (variable)																															
...																															

QuotaInformation (variable): A concatenated list of FILE_QUOTA_INFORMATION structures, as specified in [MS-FSCC] section 2.4.33.

2.2.7.4.2 Server Response

An SMB_COM_NT_TRANSACT (section 2.2.4.8) response for the NT_TRANSACT_SET_QUOTA subcommand MUST be sent by a server in reply to a client NT_TRANSACT_SET_QUOTA request when the request is successful.

The NT_TRANSACT_SET_QUOTA response is a special case of the SMB_COM_NT_TRANSACT command response. Only the NT_TRANSACT_SET_QUOTA specifics are described here.

NT_Trans_Parameters

No NT Trans parameters are returned in this response.

NT_Trans_Data

No NT Trans data is returned in this response.

Error Codes

SMB error class	SMB error code	NT status code	POSIX equivalent	Description
ERRDOS(0x01)	ERRbadfunc(0x0001)	STATUS_INVALID_DEVICE_REQUEST(0xC0000008)		Quotas are not enabled on the volume.
ERRDOS(0x01)	ERRbadfid(0x0006)	STATUS_INVALID_HANDLE(0xC0000008)	EBADF	The Fid is invalid.
ERRDOS(0x01)	ERRnoaccess(0x0005)	STATUS_ACCESS_DENIED(0xC0000022)	EPERM	Access denied.
ERRDOS(0	ERRinvalidparam(0	STATUS_INVALID_PARAMETER(0xC0000		A parameter is

SMB error class	SMB error code	NT status code	POSIX equivalent	Description
x01)	x0057)	00D)		invalid.
ERRDOS(0x01)	ERRinvalidsid(0x0539)	STATUS_INVALID_SID(0xC0000078)		The **StartSid** parameter did not contain a valid SID.
ERRSRV(0x02)	ERRerror(0x0001)	STATUS_INVALID_SMB(0x00010002)		Invalid SMB. Byte count and sizes are inconsistent.
ERRSRV(0x02)	ERRinvtid(0x0005)	STATUS_BAD_TID(0x00050002)		The TID is no longer valid.
ERRSRV(0x02)	ERRnomem(0x0008)	STATUS_INSUFF_SERVER_RESOURCES(0xC0000205)	ENOMEM	The server is out of resources.
ERRSRV(0x02)	ERRbaduid(0x005B)	STATUS_BAD_UID(0x005B0002)		The UID supplied is not known to the session.
ERRSRV(0x02)		STATUS_QUOTA_LIST_INCONSISTENT(0xC0000266)		The **SidList** parameter did not contain a valid, properly formed list.
ERRSRV(0x02)	ERRmoredata(0x00EA)	STATUS_BUFFER_OVERFLOW(0x80000005)		The number of bytes of changed data exceeded the MaxParameter Count field in the client request.
ERRHRD(0x03)	ERRdata(0x0017)	STATUS_DATA_ERROR(0xC000003E)	EIO	Disk I/O error.
ERRHRD(0x03)	ERRnowrite(0x0013)	STATUS_MEDIA_WRITE_PROTECTED(0xC00000A2)	EROFS	Attempt to modify a read-only file system.

2.2.8   Information Levels

2.2.8.1   FIND Information Level Extensions

2.2.8.1.1   SMB_FIND_FILE_BOTH_DIRECTORY_INFO Extensions

If the query being executed is a request for the enumeration of available previous versions (section 2.2.1.1.1), then the returned structure is identical to the structure that is specified in [MS-CIFS] section 2.2.8.1.7. However, the fields have a slightly different definition.

```
SMB_FIND_FILE_BOTH_DIRECTORY_INFO[SearchCount]
  {
  ULONG         NextEntryOffset;
  ULONG         FileIndex;
  FILETIME      CreationTime;
  FILETIME      LastAccessTime;
  FILETIME      LastWriteTime;
  FILETIME      LastChangeTime;
  LARGE_INTEGER EndOfFile;
  LARGE_INTEGER AllocationSize;
  ULONG         ExtFileAttributes;
  ULONG         FileNameLength;
  ULONG         EaSize;
  UCHAR         ShortNameLength;
  UCHAR         Reserved;
  WCHAR         ShortName[12];
  SMB_STRING    FileName;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
NextEntryOffset																															
FileIndex																															
CreationTime																															
...																															
LastAccessTime																															
...																															
LastWriteTime																															
...																															

Release: Tuesday, April 6, 2010

LastChangeTime			
...			
EndOfFile			
...			
AllocationSize			
...			
ExtFileAttributes			
FileNameLength			
EaSize			
ShortNameLength	Reserved	ShortName	
...			
...			
...			
...			
...			
...		FileName (variable)	
...			

FileIndex (4 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

CreationTime (8 bytes): A FILETIME time stamp of when the previous version represented by the @GMT token was created. The FILETIME format is defined in [MS-DTYP], section 2.3.1.

LastAccessTime (8 bytes): A FILETIME time stamp of when the previous version represented by the @GMT token was last accessed.

LastWriteTime (8 bytes): A FILETIME time stamp of when the previous version represented by the @GMT token last had data written to it.

LastChangeTime (8 bytes): A FILETIME time stamp of when the previous version represented by the @GMT token was last changed.

EndOfFile (8 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

AllocationSize (8 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

ExtFileAttributes (4 bytes): Extended attributes for this file that MUST be marked as a DIRECTORY.

FileNameLength (4 bytes): Length, in bytes, of the **FileName** field.

EaSize (4 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

Reserved (1 byte): An 8-bit unsigned integer used to maintain byte alignment. This field MUST be 0x00.

ShortName (24 bytes): The 8.3 name for the file formatted as @GMT~XXX where XXX is an array index value in decimal of an array of snapshots returned, starting at an array index value of zero. The **ShortName** field MUST be formatted as an array of 16-bit Unicode characters and MUST NOT be NULL terminated.

FileName (variable): An @GMT token that represents an available previous version for the file or directory.

2.2.8.1.2   SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO

The fields and encoding of the TRANS2_FIND_FIRST2 SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO Information Level response message are identical to the fields and encoding of the TRANS2_FIND_FIRST2 SMB_FIND_FILE_FULL_DIRECTORY_INFO Information Level response, as specified in [MS-CIFS] section 2.2.6.2.2, with the addition of the **FileId** field described in the list that follows the table in this section.<74>

```
SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO[SearchCount]
  {
  ULONG         NextEntryOffset;
  ULONG         FileIndex;
  FILETIME      CreationTime;
  FILETIME      LastAccessTime;
  FILETIME      LastWriteTime;
  FILETIME      LastChangeTime;
  LARGE_INTEGER EndOfFile;
  LARGE_INTEGER AllocationSize;
  ULONG         ExtFileAttributes;
  ULONG         FileNameLength;
  ULONG         EaSize;
  LARGE_INTEGER FileID;
  SMB_STRING    FileName;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
NextEntryOffset																															
FileIndex																															
CreationTime																															
...																															
LastAccessTime																															
...																															
LastWriteTime																															
...																															
LastChangeTime																															
...																															
EndOfFile																															
...																															
AllocationSize																															
...																															
ExtFileAttributes																															
FileNameLength																															
EaSize																															
FileId																															
...																															

FileName (variable)
...

FileIndex (4 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

EndOfFile (8 bytes): This **LARGE_INTEGER** field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

AllocationSize (8 bytes): This **LARGE_INTEGER** field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

ExtFileAttributes (4 bytes): Extended attributes for this file that MUST be marked as a DIRECTORY.

FileNameLength (4 bytes): The length, in bytes, of the **FileName** field.

EaSize (4 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

FileId (8 bytes): A **LARGE_INTEGER** that serves as an internal file system identifier. This number MUST be unique for each file on a given volume. If a remote file system does not support unique FileId values, then the **FileId** field MUST be set to zero.

2.2.8.1.3   SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO

The fields and encoding of the TRANS2_FIND_FIRST2 SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO Information Level response message are identical to the fields and encoding of SMB_FIND_FILE_BOTH_DIRECTORY_INFO Information Level, as specified in [MS-CIFS], section 2.2.6.2.2, with the addition of the **Reserved2** and **FileId** fields described in the list that follows the table.<75>

```
SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO[SearchCount]
  {
  ULONG         NextEntryOffset;
  ULONG         FileIndex;
  FILETIME      CreationTime;
  FILETIME      LastAccessTime;
  FILETIME      LastWriteTime;
  FILETIME      LastChangeTime;
  LARGE_INTEGER EndOfFile;
  LARGE_INTEGER AllocationSize;
  ULONG         ExtFileAttributes;
  ULONG         FileNameLength;
  ULONG         EaSize;
  UCHAR         ShortNameLength;
  UCHAR         Reserved;
  WCHAR         ShortName[12];
  USHORT        Reserved2;
  LARGE_INTEGER FileID;
  SMB_STRING    FileName;
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
NextEntryOffset																																
FileIndex																																
CreationTime																																
...																																
LastAccessTime																																
...																																
LastWriteTime																																
...																																
LastChangeTime																																
...																																
EndOfFile																																
...																																
AllocationSize																																
...																																
ExtFileAttributes																																
FileNameLength																																
EaSize																																
ShortNameLength								Reserved								ShortName																
...																																

...	
...	
...	
...	
...	Reserved2
FileId	
...	
FileName (variable)	
...	

FileIndex (4 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

AllocationSize (8 bytes): This **LARGE_INTEGER** field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

ExtFileAttributes (4 bytes): This field represents the extended attributes for this file that MUST be marked as a DIRECTORY.

EaSize (4 bytes): This field MUST be set to zero when sending a response and MUST be ignored when the client receives this message.

Reserved (1 byte): An 8-bit unsigned integer that is used to maintain 64-bit alignment. This member MUST be 0x00.

Reserved2 (2 bytes): A 16-bit unsigned integer that is used to maintain 64-bit alignment. This member MUST be 0x0000.

FileId (8 bytes): A **LARGE_INTEGER** that serves as an internal file system identifier. This number MUST be unique for each file on a given volume. If a remote file system does not support unique FileId values, then the **FileId** field MUST be set to zero.

2.2.8.2   QUERY_FS Information Level Extensions

2.2.8.2.1   SMB_QUERY_FS_ATTRIBUTE_INFO

For this Information Level, the server SHOULD check the **FileSystemAttributes** field and remove the attribute flags that are not supported by the underlying object store before sending the response to the client.<76>

2.2.8.3 QUERY Information Level Extensions

No new SMB-specific Information Levels are specified for these extensions.

2.2.8.4 SET Information level Extensions

No new SMB-specific Information Levels are specified for these extensions.<77>

Release: Tuesday, April 6, 2010

3   Protocol Details

An SMB implementation MUST implement CIFS, as specified by section 3 of the [MS-CIFS] specification.

3.1   Common Details

3.1.1   Abstract Data Model

This section specifies a conceptual model of possible data organization that an implementation maintains in order to participate in this protocol. The described organization is provided to explain how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with what is described in this document.

The following elements extend the global abstract data model that is specified in [MS-CIFS] section 3.1.1.

3.1.1.1   Global

There are no global parameters defined as common to both client and server.

3.1.2   Timers

There are no timers common to both client and server.

3.1.3   Initialization

No new common variables are defined in this document.

3.1.4   Higher-Layer Triggered Events

3.1.4.1   Sending Any Message

Processing of any message is handled as specified in [MS-CIFS] section 3.1.4.1 with the following additions: The MD5 algorithm, as specified in [RFC1321], MUST be used to generate a hash of the SMB message (from the start of the SMB_Header), which is defined as follows.<78>

```
IF ( Connection.SigningChallengeResponse != NULL ) THEN
   CALL MD5Init( md5context )
   CALL MD5Update( md5context, Connection.SigningSessionKey )
   CALL MD5Update( md5context,
                   Connection.SigningChallengeResponse )
   CALL MD5Update( md5context, SMB message )
   CALL MD5Final( digest, md5context )
ELSE
   CALL MD5Init( md5context )
   CALL MD5Update( md5context, Connection.SigningSessionKey )
   CALL MD5Update( md5context, SMB message )
   CALL MD5Final( digest, md5context )
END IF
SET the signature TO the first 8 bytes of the digest
```

The resulting 8-byte signature MUST be copied into the **SecuritySignature** field of the SMB header, after which the message can be transmitted.

3.1.5 Processing Events and Sequencing Rules

3.1.5.1 Receiving Any Message

Processing of any message is handled as specified in [MS-CIFS] section 3.1.5.1 with the following additions:

The MD5 algorithm, as specified in [RFC1321], MUST be used to generate a hash of the SMB message (from the start of the SMB header), and SHOULD be used as follows.<79>

```
IF ( Connection.SigningChallengeResponse != NULL ) THEN
   CALL MD5Init( md5context )
   CALL MD5Update( md5context, Connection.SigningSessionKey )
   CALL MD5Update( md5context,
                   Connection.SigningChallengeResponse )
   CALL MD5Update( md5context, SMB message )
   CALL MD5Final( digest, md5context )
ELSE
   CALL MD5Init( md5context )
   CALL MD5Update( md5context, Connection.SigningSessionKey )
   CALL MD5Update( md5context, SMB message )
   CALL MD5Final( digest, md5context )
END IF
SET the signature TO the first 8 bytes of the digest
```

The resulting 8-byte signature is compared with the original value of the **SMB_Header.SecuritySignature** field. If the signature received with the message does not match the signature that is calculated, then the message MUST be discarded and no further processing on it is done. The receiver MAY also terminate the connection by disconnecting the underlying transport connection and cleaning up any state associated with the connection.

3.1.6 Timer Events

There are no timers common to both client and server.

3.1.7 Other Local Events

There are no local events common to both client and server.

3.2 Client Details

3.2.1 Abstract Data Model

This section specifies a conceptual model of possible data organization that an implementation maintains in order to participate in this protocol. The described organization is provided to explain how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with what is described in this document.

The following elements extend the client abstract data model specified in [MS-CIFS] section 3.2.1.

3.2.1.1 Global

Client.MessageSigningPolicy: A state that determines whether or not this node signs messages. This parameter has four possible values:

- *Required* -- Message signing is required. Any connection to a server that does not use signing MUST be disconnected.

- *Preferred* -- Message signing is enabled. If the other node enables or requires signing, signing MUST be used.

- *Declined* -- Message signing is disabled unless the other party requires it. If the other party requires message signing, it MUST be used. Otherwise, message signing MUST NOT be used.

- *Disabled* -- Message signing is disabled. Message signing MUST NOT be used.

Client.SupportsExtendedSecurity: A flag that indicates whether the client supports Generic Security Services (GSS), as specified in [RFC2743], for selecting the authentication protocol.

3.2.1.2 Per SMB Connection

Client.Connection.GSSNegotiateToken: A byte array that contains the token received during an extended security negotiation and that is remembered for authentication.

Client.Connection.ServerGUID: A GUID generated by the server to uniquely identify this server.

3.2.1.3 Per SMB Session

Client.Session.AuthenticationState: A session can be in one of three states:

- InProgress -- A session setup (an extended SMB_COM_SESSION_SETUP_ANDX exchange as described in section 3.2.4.2.4.1) is in progress for this session.

- Valid -- A session setup exchange has successfully completed; the session is valid and a UID for the session has been assigned by the server.

- Expired -- The Kerberos ticket for this session has expired and the session needs to be re-established.

Client.Session.SessionKeyState: The session key state. This can be either Unavailable or Available.

Client.Session.UserCredentials: An opaque implementation-specific entity that identifies the credentials that were used to authenticate to the server.

3.2.1.4 Per Tree Connect

Client.TreeConnect.GuestMaximalShareAccessRights: The **GuestMaximalShareAccessRights** value as returned in the SMB_COM_TREE_CONNECT_ANDX server response (section 2.2.4.7.2).

Client.TreeConnect.MaximalShareAccessRights: The **MaximalShareAccessRights** value as returned in the SMB_COM_TREE_CONNECT_ANDX server response (section 2.2.4.7.2).

3.2.2 Timers

There are no new client timers other than those specified in [MS-CIFS].

3.2.3 Initialization

Initialization of the following additional parameters is required beyond that specified in [MS-CIFS].

The following values MUST be initialized at system startup:

- **Client.MessageSigningPolicy** and **Client.SupportsExtendedSecurity** MUST be set based on system policy.<80> The value of this is not constrained by the values of any other policies.

When an SMB connection is established, the following values MUST be initialized:

- **Client.Connection.GSSNegotiateToken** is set to an empty array.

- **Client.Connection.ServerGUID** is set to the GUID of the server.

When an SMB session is established on an SMB connection, the following value MUST be initialized:

- **Client.Session.AuthenticationState** MUST be set to *InProgress*.

- **Client.Session.SessionKeyState** MUST be set to *Unavailable*.

- **Client.Session.UserCredentials** MUST be empty.

All other values are initialized as specified in [MS-CIFS] section 3.2.3.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Sending Any Message

The following global details are presented to a client that sends any message in addition to what is specified in [MS-CIFS] section 3.2.4.1.

3.2.4.1.1 Scanning a Path for a Previous Version Token

The application requests a previous version of a file by placing a time indicator in the path as a directory element, as specified in section 2.2.1.1.1. For any path-based operation (for example,SMB_COM_NT_CREATE_ANDX) the client SHOULD scan the file path being provided by the application for an appropriately formatted @GMT token.

If a previous version token is present in the pathname as a directory element or a final target, the client SHOULD<81> set the SMB_FLAGS2_REPARSE_PATH flag in the SMB header of the request.

3.2.4.2 Application Requests Connecting to a Share

3.2.4.2.1 Connection Establishment

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.2.1.

3.2.4.2.2 Dialect Negotiation

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.2.2, with the following additions:

Require Message Signing

If the client's **Client.MessageSigningPolicy** is "Required", the client SHOULD<82> set the SMB_FLAGS2_SMB_SECURITY_SIGNATURE_REQUIRED bit in the **Flags2** field of the SMB_COM_NEGOTIATE request SMB header to indicate that the client MUST refuse to connect if signing is not used.

3.2.4.2.3   Capabilities Negotiation

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.2.3, with the addition that the new capabilities flags (specified in section 2.2.4.5.2) are also to be considered in the list of possible capabilities.

3.2.4.2.4   User Authentication

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.2.4, with the following additions:

If **Client.Connection.ShareLevelAccessControl** is FALSE:

If one or more connections to the server exist for **Client.ConnectionTable[ServerName]**, then the client MUST traverse the Connections and, for each, search the **Client.Connection.SessionTable** for a Session that corresponds to the user that establishes the connection to the share. The client MUST search for a valid Session by either a security context or a UID representing the user.

- If a Connection with an existing Session for this user is found, then the client MUST reuse the connection and continue processing.

- If there is no connection with a Session for this user, then the client MAY<83> select any connection to the server and continue processing.

If no valid Connections exist, or if the client chooses not to reuse an existing connection, a new SMB connection MUST be established and the client MUST attempt to establish a new authenticated Session for the user. The user's credentials, --typically a username or principal and an associated password or password hash,-- MUST be stored in **Client.Session.UserCredentials**.

Signing

The client-global **Client.MessageSigningPolicy** MUST be compared against the selected **Client.Connection.ServerSigningState**, as per the following table. If the result is *Blocked*, the underlying transport connection SHOULD be closed.<84>

Client signing state	Server signing state			
	Disabled	Declined	Enabled	Required
Disabled	Message Unsigned	Message Unsigned	Message Unsigned	Blocked
Declined	Message Unsigned	Message Unsigned	Message Unsigned	Message Signed
Enabled	Message Unsigned	Message Unsigned	Message Signed	Message Signed
Required	Blocked	Message Signed	Message Signed	Message Signed

Extended Security

If **Client.Connection.ServerCapabilities** has the CAP_EXTENDED_SECURITY flag set (which indicates that the server supports extended security), then the client MUST construct an SMB_COM_SESSION_SETUP_ANDX request, as specified in section 2.2.4.6.1.

The client MUST do one of the following:

- Pass the **Client.Connection.GSSNegotiateToken** (if valid) to the configured GSS authentication mechanism to obtain a GSS output token for the authentication protocol exchange.<85>

  -OR-

- Choose to ignore the **Client.Connection.GSSNegotiateToken** received from the server, and give an empty input token to the configured GSS authentication protocol to obtain a GSS output token for the authentication protocol exchange.

In either case, the client MUST initiate the GSS authentication protocol via the GSS_Init_sec_context() interface, as specified in [RFC2743], passing in the input Client.Connection.GSSNegotiateToken as previously described, along with the credentials from **Client.Session.UserCredentials**. The client MUST set the MutualAuth and Delegate options which are specified in [MS-KILE] section 3.2.1.<86>

The client MUST then create an SMB_COM_SESSION_SETUP_ANDX request (section 2.2.4.6.1) message. The client MUST set CAP_EXTENDED_SECURITY in the **Capabilities** field, SMB_FLAGS2_EXTENDED_SECURITY in the **SMB_Header.Flags2** field, the **SMB_Data.Bytes.SecurityBlob** field to the GSS output token generated by the GSS_Init_sec_context() interface, and the **SMB_Parameters.Words.SecurityBlobLength** to the length of the GSS output token. The **UID** field in the SMB header MUST be set to zero.

3.2.4.2.4.1 Sequence Diagram

For the user to be successfully authenticated and to establish a session, the client MUST follow a security negotiation scheme that can involve one or more roundtrips of SMB_COM_SESSION_SETUP_ANDX request and response. In each roundtrip, the server and client exchange security tokens. The exchange of security tokens MUST continue until either the client or the server determines that authentication has failed or both sides decide that authentication is complete. If authentication fails, then the client drops the connection and indicates the error (see the following diagram for details). If authentication succeeds, then the application protocol can be assured of the identity of the participants as far as the supporting authentication protocol can accomplish.

In the sequence diagram that follows, requests with straight line arrows stand for the requests that the client MUST send. Requests with dotted line arrows stand for the requests the client may send. The server MUST respond to each client request that it receives.
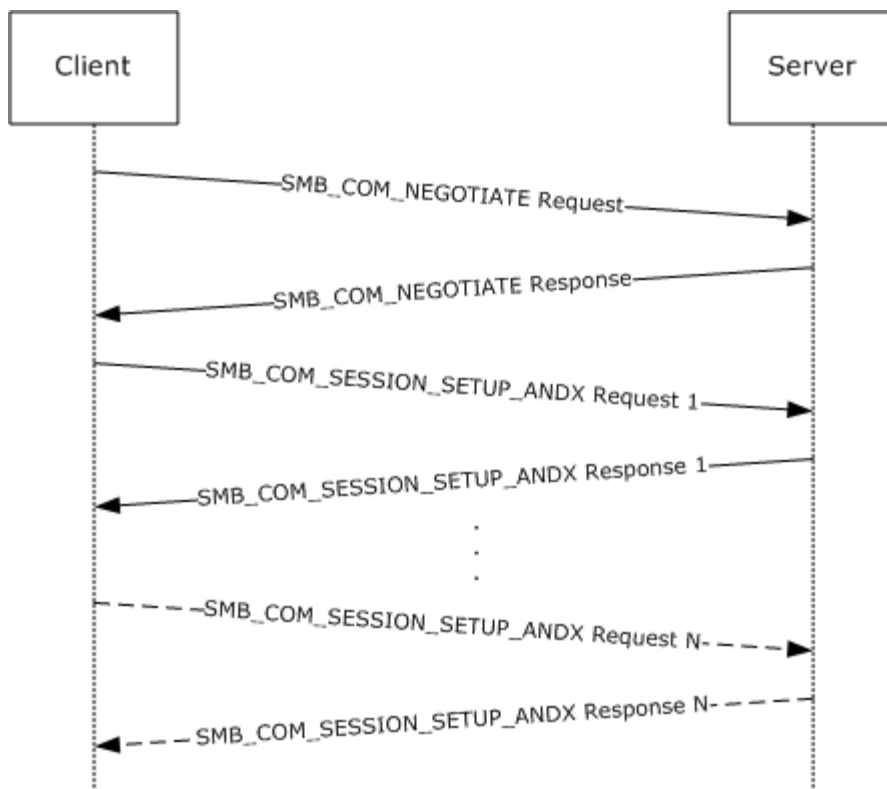
Figure 2: User authentication and session establishment sequence

The diagram illustrates the sequence of events during the protocol negotiation and session establishment process. After the initial SMB_COM_NEGOTIATE command exchange has been completed, the SMB_COM_NEGOTIATE exchange MUST NOT be repeated over the same SMB connection. The parameters returned in the SMB_COM_NEGOTIATE response MUST be used when creating new sessions over the same connection.

Session Setup Roundtrip

The SMB_COM_NEGOTIATE Server response is processed as described in [MS-CIFS] section 3.2.5.2. The protocol extensions in this document apply only to the NT LM 0.12 dialect of SMB. For further information on SMB dialects, see [MS-CIFS] section 1.7.

If the NT LM 0.12 dialect is successfully negotiated, then the SMB client examines the Capabilities field in the SMB_COM_NEGOTIATE server response (section 2.2.4.5.2). If the CAP_EXTENDED_SECURITY bit is clear (0x00000000), then the SMB server does not support extended security. In order for authentication to proceed, the SMB client MUST build a non-extended SMB_COM_SESSION_SETUP_ANDX request, and MUST set the **WordCount** field to 0x0d. Authentication then proceeds as described in [MS-CIFS] section 3.2.4.2.4.

If the CAP_EXTENDED_SECURITY bit is set (0x80000000), then the SMB server does support extended security. The SMB client MUST build an SMB_COM_SESSION_SETUP_ANDX request in the extended form, as specified in section 2.2.4.6.1. The request is sent to the SMB server, and the server builds an extended SMB_COM_SESSION_SETUP_ANDX server response (section 2.2.4.6.2.). The security BLOB in the session setup response is built as specified in [RFC2743].

Upon receiving the extended SMB_COM_SESSION_SETUP_ANDX server response (section 2.2.4.6.2), the SMB client invokes the local security package to determine whether the session setup request SHOULD be completed, aborted, or continued. A completed session indicates that the server has enough information to establish the session. An aborted session indicates that the server cannot proceed with the session setup because of an error in the information presented by the client, or otherwise. If the session setup has to be continued, the security package on the client and/or server requires an additional roundtrip before the session setup can be established. This is especially true of new security packages that support mutual authentication between the client and server.

In the case of extended security, the SMB protocol does not make the distinction between NTLM and Kerberos;therefore, the sequence defined previously in this section is the same in both cases.

The protocol terminates after one round of session setup request and response is exchanged in the case of NTLM. It can require multiple trips in the case of other security packages before the session setup is established.

In the case of additional roundtrips, each roundtrip MUST consist of one SMB_COM_SESSION_SETUP_ANDX client request and one SMB_COM_SESSION_SETUP_ANDX server response. In the sequence diagram, this is represented in the horizontal dotted line that symbolizes additional round trips until the final round trip, which is represented as SMB_COM_SESSION_SETUP_ANDX Client Request N and SMB_COM_SESSION_SETUP_ANDX Server Response N, where N is a number larger than 1.

All additional SMB session setup round trips follow the same sequence details as Session Setup Round Trip, as described earlier in this topic.<87>

3.2.4.2.5 Connecting to the Share (Tree Connect)

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.2.5, with the following additions:

If a tree connect is already established to the target share in **Client.Connection.TreeConnectTable**, then it SHOULD<88> be reused. If not, then the client creates an SMB_COM_TREE_CONNECT_ANDX request, as specified in section 2.2.4.7.1, to connect to the target share.

Session Key Protection

The client can request **Client.Session.SessionKey** protection by setting the TREE_CONNECT_ANDX_EXTENDED_SIGNATURES flag in the **Flags** field of the SMB_COM_TREE_CONNECT_ANDX request to TRUE.<89>

Extended Information Response

The client SHOULD<90> request extended information in the response by setting the TREE_CONNECT_ANDX_EXTENDED_RESPONSE flag in the **Flags** field of the SMB_COM_TREE_CONNECT_ANDX request to TRUE, as defined in section 2.2.4.3.2.

The client sends this message to the server.

3.2.4.3 Application Requests Opening a File

The processing of this event is handled as specified in [MS-CIFS] sections 3.2.4.5 and 3.2.4.6, with the following additions:

The application can request that additional information be returned for the Open, in particular maximal access information. The client can issue either an SMB_COM_NT_CREATE_ANDX request or an SMB_COM_OPEN_ANDX request to make use of these extensions, as specified in section 3.2.4.3.1.<91>

3.2.4.3.1 SMB_COM_NT_CREATE_ANDX Request

To access these extensions, the application can also provide:

- **RequestExtendedResponse:** A BOOLEAN. If TRUE, then it indicates that the application is requesting a server to send an extended response.

If the application is requesting an extended server response, then the client MUST set the NT_CREATE_REQUEST_EXTENDED_RESPONSE flag in the **SMB_Parameters.Flags** field of the request.

3.2.4.3.2 SMB_COM_OPEN_ANDX Request (deprecated)

To access these extensions, the application can also provide:

- **RequestExtendedResponse:** A BOOLEAN. If TRUE, then it indicates that the application is requesting a server to send an extended response.

If the application is requesting an extended server response, then the client SHOULD<92> set the SMB_OPEN_EXTENDED_RESPONSE flag in the **SMB_Parameters.Flags** field of the request.

3.2.4.4 Application Requests Reading from a File, Named Pipe, or Device

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.13, with the following additions:

The SMB_COM_READ_ANDX command request has been extended as specified below.

Timeout_or_MaxCountHigh

The **Timeout** field specified in [MS-CIFS] section 2.2.4.42.1 has been extended to be the **Timeout_or_MaxCountHigh** field. This field is treated as a union of a 32-bit **Timeout** field and a 16-bit unsigned integer named **MaxCountHigh**, as specified in section 2.2.4.2.1.

- For pipe reads, the client MUST use **Timeout_or_MaxCountHigh** as the **Timeout** field. The client MUST set the **Timeout** field either to a time-out value in milliseconds, or to 0xFFFFFFFF.<93> The latter value indicates to the server that the operation MUST NOT time out.

- For file reads, the client MUST use this as the **MaxCountHigh** field. If the count of bytes to read is larger than 0xFFFF bytes in length, then the client MUST use the **MaxCountHigh** field to hold the two most significant bytes of the count, thus allowing for a 32-bit read count. If the read count is not larger than 0xFFFF, then the client MUST set **MaxCountHigh** to zero.

3.2.4.4.1 Large Read Support

If the CAP_LARGE_READX bit is set in **Client.Connection.ServerCapabilities**, then the client is allowed to issue a read of a size larger than **Client.Connection.MaxBufferSize** using an SMB_COM_READ_ANDX request. Otherwise, the client MUST split the read into multiple requests in order to retrieve the entire amount of data.<94>

- If a large read is being issued and the object being read is not a file, then the count of bytes to read MUST be placed into the **MaxCountOfBytesToReturn** field. This field is a 16-bit unsigned integer; therefore, the maximum count of bytes that can be read is 0xFFFF bytes (64K - 1 byte).

- If a large read is being issued and the object being read is a file, then the two least significant bytes of the count of bytes to read MUST be placed into the **MaxCountOfBytesToReturn** field and the two most significant bytes of the count MUST be placed into the **MaxCountHigh** field.

3.2.4.5 Application Requests Writing to a File, Named Pipe, or Device

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.14 with the following additions:

The SMB_COM_WRITE_ANDX command request has been extended as specified in section 3.2.5.8.

Large Write Support

If the CAP_LARGE_WRITEX bit is set in **Client.Connection.ServerCapabilities**, then the client is allowed to issue a write of a size larger than **Client.Connection.MaxBufferSize** using an SMB_COM_WRITE_ANDX request. Otherwise, it MUST split the write into multiple requests to write the entire amount of data.

If the count of bytes to be written is greater than or equal to 0x00010000 (64K), then the client MUST set the two least significant bytes of the count in the **DataLength** field of the request and the two most significant bytes of the count in the **DataLengthHigh** field.

3.2.4.6 Application Requests a Directory Enumeration

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.27, with the following additions:

The TRANS2_FIND_FIRST2 subcommand request has been extended as specified below.

New Information Levels

To request the new Information Levels specified in section 2.2.6.1.1, the client MUST set the **InformationLevel** field of the TRANS2_FIND_FIRST2 request to the corresponding Information Level.

Enumerating Previous Versions

An application is allowed to request an enumeration of available previous versions of a file or directory using a TRANS2_FIND_FIRST2 request (see section 2.2.1.1.1). To do this, the request MUST have the @GMT token wildcard, @GMT-*, as part of the search pattern in the **FileName** field and it MUST use the SMB_FIND_FILE_BOTH_DIRECTORY_INFO Information Level. This extension is not available for Information Levels other than SMB_FIND_FILE_BOTH_DIRECTORY_INFO. The client MAY<95> fail such requests or simply send the requests to the server. Because it is a path-based operation, this request follows the same previous version token parsing rules as specified in section 3.2.4.1.1.

The message is sent to the server.

3.2.4.7 Application Requests Querying File Attributes

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.11, with the following additions:

Pass-through Information Levels

The extension adds support for pass-through Information Levels, as defined in section 2.2.2.3. If the CAP_INFOLEVEL_PASSTHRU bit in **Client.Connection.ServerCapabilities** is set, and the application query is for a file information class (see [MS-FSCC] section 2.4 for possible values), then the client MAY increment the Information Level value by SMB_INFO_PASSTHROUGH (0x03e8) and place the resulting value in the **InformationLevel** field of a TRANS2_QUERY_FILE_INFORMATION or TRANS2_QUERY_PATH_INFORMATION request. Otherwise, it MUST choose an appropriate, existing SMB Information Level and reconstruct the information required by the application.<96> If an appropriate SMB Information Level is not available, then the server MUST fail the request with STATUS_NOT_SUPPORTED.

File Streams

A client MAY send a TRANS2_QUERY_FILE_INFORMATION subcommand of the SMB_COM_TRANSACTION2 request to the server with the **InformationLevel** field set to SMB_QUERY_FILE_STREAM_INFO (see [MS-CIFS] section 2.2.6.6.2). If the FID field in the client request is on an SMB share that does not support streams, then the server SHOULD<97> fail the request with STATUS_NOT_SUPPORTED.

A client MAY send a TRANS2_QUERY_PATH_INFORMATION subcommand of the SMB_COM_TRANSACTION2 request to the server with the **InformationLevel** field set to SMB_QUERY_FILE_STREAM_INFO (see [MS-CIFS] section 2.2.6.6.2). If the **FileName** field in the client request is on an SMB share that does not support streams, then the server SHOULD<98> fail the request with STATUS_NOT_SUPPORTED.

Previous Version Tokens

Because the TRANS2_QUERY_PATH_INFORMATION subcommand request is a path-based operation, the path SHOULD be scanned for previous version tokens by the client, as specified in section 3.2.4.1.1.

3.2.4.8   Application Requests Setting File Attributes

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.12, with the following additions:

Pass-Through Information Levels

The extension adds support for pass-through **Information Levels**, as defined in section 2.2.2.3. If the CAP_INFOLEVEL_PASSTHRU bit in **Client.Connection.ServerCapabilities** is set and the application query is for a file information class (see [MS-FSCC] section 2.4 for possible values), then the client MAY increment the level value by SMB_INFO_PASSTHROUGH (0x03e8) and place the resulting value in the **InformationLevel** field of a TRANS2_SET_FILE_INFORMATION or TRANS2_SET_PATH_INFORMATION request. The serialized native structure is placed in the Trans2_Data block of the request and the **SMB_Parameters.TotalDataCount** is set to the length of this buffer. Otherwise, it MUST choose an appropriate, existing SMB Information Level and reconstruct the information required by the application.<99> If an appropriate SMB Information Level is not available, then the server MUST fail the request with STATUS_NOT_SUPPORTED.

Previous Version Tokens

Because the TRANS2_SET_PATH_INFORMATION subcommand request is a path-based operation, the path SHOULD be scanned for previous version tokens by the client, as specified in section 3.2.4.1.1.

3.2.4.9   Application Requests Querying File System Attributes

The processing of this event is handled as specified in [MS-CIFS] section 3.2.4.25, with the following additions:

Support of pass-through Information Levels for the querying file system attributes through the use of the TRANS2_QUERY_FS_INFORMATION subcommand is defined in section 2.2.2.3. If the CAP_INFOLEVEL_PASSTHRU bit in **Client.Connection.ServerCapabilities** is set and the application query is for a file information class (see [MS-FSCC] section 2.5 for possible values), then the client MAY increment the Information Level value by SMB_INFO_PASSTHROUGH (0x03e8) and place the resulting value in the **InformationLevel** field of a TRANS2_QUERY_FS_INFORMATION request. Otherwise, it MUST choose an appropriate, existing SMB Information Level and reconstruct the information that is required by the application.<100> If an appropriate SMB Information Level is not available, then the server MUST fail the request with STATUS_NOT_SUPPORTED.

3.2.4.10   Application Requests Setting File System Attributes

The application MUST provide the following:

- An **Open** that identifies a file on the file system that will have its attributes set.

- The Information Level that defines the format of the data to set. Valid Information Levels are specified in [MS-FSCC] section 2.5.

- A buffer that contains the attribute data to be set on the server. The buffer is formatted as specified in the subsection of [MS-FSCC] section 2.5 that corresponds to the Information Level supplied by application.

This operation supports the use of pass-through Information Levels only. If the CAP_INFOLEVEL_PASSTHRU flag in **Client.Connection.ServerCapabilities** is not set, then the client MUST fail the request with STATUS_NOT_SUPPORTED.

The client MUST construct a TRANS2_SET_FS_INFORMATION subcommand request, as specified in section 2.2.6.4.1.

The client MUST use **Open.TreeConnect** and **Open.Session** to send the request to the server. The request MUST be sent to the server, as specified in section 3.2.4.1.

3.2.4.11   Application Requests Sending an I/O Control to a File or Device

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.22, with the following additions:

A list of **FSCTLs** is specified in [MS-FSCC] section 2.3. Three I/O control codes specific to the extension are described in the following subsections.

3.2.4.11.1   Application Requests Enumerating Available Previous Versions

An application can request that a client retrieve an enumeration of available previous version time stamps for a share by issuing the FSCTL_SRV_ENUMERATE_SNAPSHOTS control code, as specified in section 2.2.7.2.1.<101> The request is sent to the server.

3.2.4.11.2   Application Requests a Server-Side Data Copy

An outline of the steps taken for a server-side data copy of an entire file follows. Similar steps can be used to copy partial ranges within a file. The client issues a query for the copychunk resume key

on the source file. The client MUST then pass this value into each iteration of the copychunk FSCTL that copies a given range on the server side. The client continues to issue these copy requests for subsequent data ranges until the entire file has been copied.
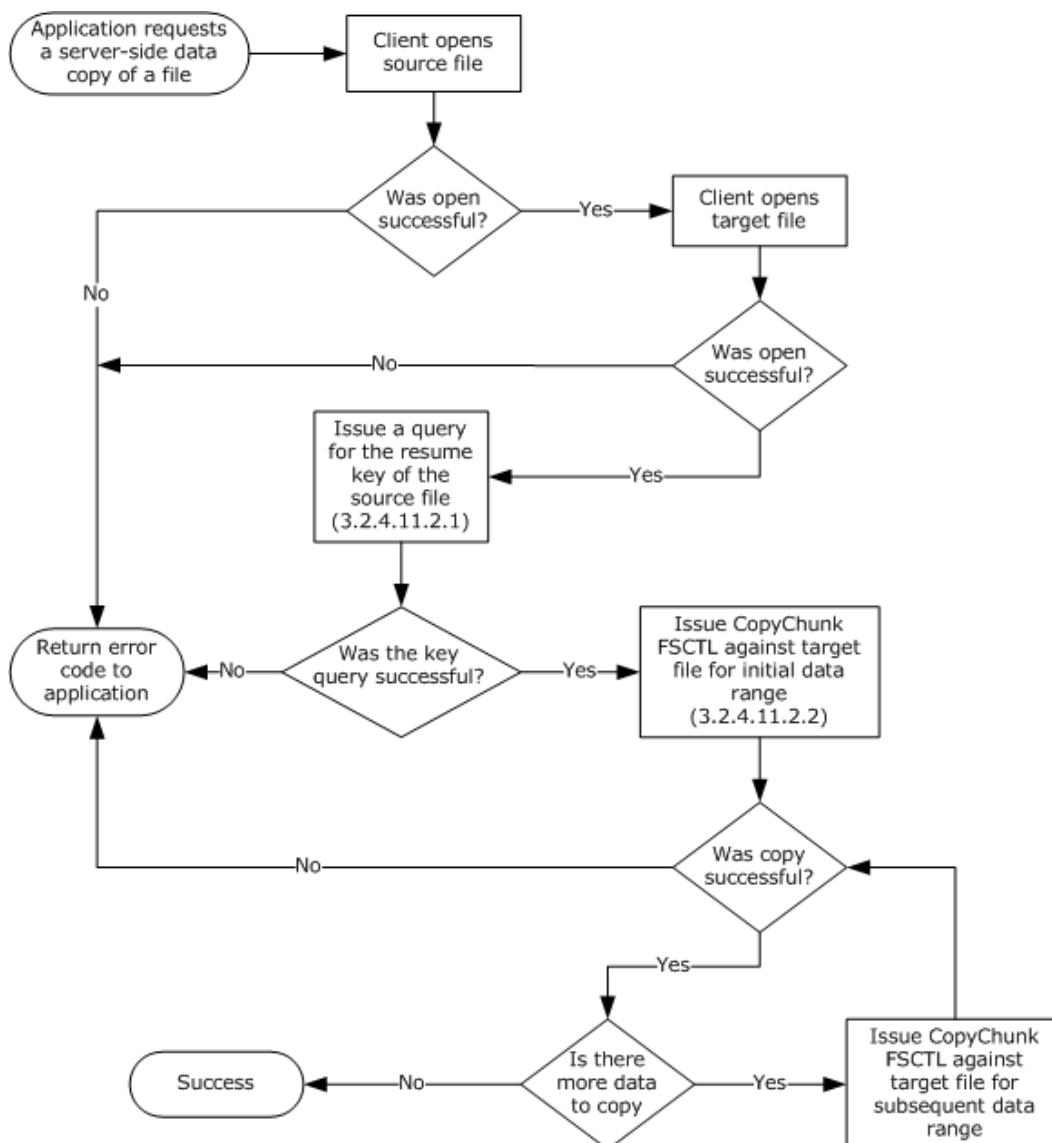
Figure 3: Server-side data copy of an entire file

3.2.4.11.2.1 Issue a Query for the Copychunk Resume Key of the Source File

To request a copychunk resume key for an open file, the client constructs an NT_TRANSACT_IOCTL FSCTL_SRV_REQUEST_ RESUME_KEY request, as specified in section 2.2.7.2.1. The Fid of the source file (**Client.Open.FID**) is placed in the **FID** field of the client request along with the FSCTL_SRV_REQUEST_RESUME_KEY function code. No **NT_Trans_Data** block is required. The request is sent to the server.

3.2.4.11.2.2 Issue CopyChunk FSCTL Against Target File for Initial/subsequent Data Range

To request a server-side copy of a data range, the client constructs an NT_TRANSACT_IOCTL FSCTL_SRV_COPYCHUNK request, as specified in section 2.2.7.2.1. The copychunk resume key retrieved for the source file and the range of data blocks to be copied are placed in the **NT_Trans_Data.Data** buffer of the transaction request. The Fid of the destination file (**Client.Open.FID**) is placed in the **FID** field of the request along with the FSCTL_SRV_COPYCHUNK function code. The request is sent to the server.

3.2.4.12 Application Requests Querying of DFS Referral

Processing of this event is handled as specified in [MS-CIFS] section 3.2.4.44.<102>

3.2.4.13 Application Requests Querying User Quota Information

The application MUST provide:

- A valid **Open** to a file or directory on a share. The quota information on the object store that underlies the file or directory is the quota information to be queried.

- A buffer to receive the quota information and the maximum number of bytes to receive.

- **RestartScan:** A **BOOLEAN** that indicates whether or not a scan on the volume is to be restarted.

- **ReturnSingleEntry:** A **BOOLEAN**. If TRUE, then the server MUST return a single user quota information entry.

- A security identifier (SID) list, a start SID, or no SID. If the application provides both an SID list and a start SID, then the client MUST fail the request with STATUS_INVALID_PARAMETER.

The client MUST construct an NT_TRANSACT_QUERY_QUOTA subcommand request, as specified in section 2.2.7.3.1, with the following additional requirements:

- **NT_Trans_Parameters.FID** MUST be set to the Fid of the application-supplied Open.

- **NT_Trans_Parameters.ReturnSingleEntry** MUST be set to the value of the application-supplied **ReturnSingleEntry** BOOLEAN.

- **NT_Trans_Parameters.RestartScan** MUST be set to the value of the application-supplied RestartScan BOOLEAN.

- The **NT_Trans_Data.SidList** field is set to either the application-supplied SID list or start **SID**. If neither were supplied, then this field is not included.

- The NT_Trans_Parameters fields of **SidListLength**, **StartSidLength**, and **StartSidOffset** MUST be set according to the following rules:

  - If the application provides a SID list (a list of SIDs that represents users whose quota information is to be queried), then the client MUST set the **SidList** field of the request to this list and set **SidListLength** to the length of the list. In this case, **StartSidLength** and **StartSidOffset** MUST be zero.

  - If the application provides a start SID(a single SID that indicates to the server where to start user quota information enumeration), then the client MUST set **StartSidLength** to the length

of the SID and **StartSidOffset** to the offset in bytes of the **NT_Trans_Data.SidList** field relative to the start of the SMB header. In this case, **SidListLength** MUST be zero.

- If the application does not provide a SID list or a start SID, then **StartSidLength**, **StartSidOffset**, and **SidListLength** MUST be zero. If the application provides both a SID list and a start SID, then the client MUST fail the request with STATUS_INVALID_PARAMETER.

The request is sent to the server.

3.2.4.14 Application Requests Setting User Quota Information

The application MUST provide:

- A valid Open to a file or directory on a share . The quota information of the object store that underlies the file or directory is the quota information to be modified.

- A list of the security identifiers (SIDs, representing users) and their associated quota information to be set.

The client MUST construct an NT_TRANSACT_SET_QUOTA subcommand request, as specified in section 2.2.7.4.1, with the following additional parameters:

- **NT_Trans_Parameters.FID** MUST be set to the Fid of the application-supplied Open.

- The application-supplied list of SIDs and associated user quota information MUST be set as the contents of the NT_Trans_Data block, as specified in section 2.2.7.4.1.

The client sends the request to the server.

3.2.4.15 Application Requests the Session Key for a Connection

An application provides one of the following:

- An **Open** to a file or pipe.

  OR

- An SMB session that identifies an authenticated connection to the server.

If an Open was supplied by the caller, then **Client.Open.Session** MUST be used.

If a **Session** is found, but **Client.Session.AuthenticationState** is not *Valid*, then an implementation-specific error MUST be returned to the caller that indicates that the session key is not available.

If a session is found and **Client.Session.AuthenticationState** is *Valid*, but **Client.Session.SessionKeyState** is *Unavailable*, then the request MUST be failed with STATUS_ACCESS_DENIED.

If a session is found, **Client.Session.AuthenticationState** is *Valid*, and **Client.Session.SessionKeyState** is *Available*, then the 16-byte **Client.Session.SessionKey** MUST be returned to the calling application.

3.2.5 Processing Events and Sequencing Rules

3.2.5.1 Receiving Any Message

In addition to the global processing rules for a client that receives any message, as specified in [MS-CIFS] section 3.2.5.1, the following processing rules apply to the extensions presented in this document.

Signing

If a message is received and Client.Connection.IsSigningActive is TRUE for the connection, then the signature MUST be verified, as specified in section 3.1.5.1, unless the message is an oplock break, as specified in [MS-CIFS] section 3.2.5.42.

The client is responsible for providing the expected sequence number for signature verification. The sequence number for the incoming response is determined by what was stored in the **Client.Connection.ClientResponseSequenceNumber** table. The client MUST look up the expected sequence number in that table based on the PID and **MID** of the response. The client uses **Client.Connection.ClientResponseSequenceNumber[PID,MID]**as the sequence number in signature verification, as specified in section 3.1.5.1. If signature verification fails, then the message MUST be discarded and not processed. The client SHOULD<103> disconnect the underlying connection and tear down all states associated with this connection.

Session Expiration and Re-authentication

If the request passed a valid authenticated session identifier in the **SMB_Header.UID** field and the status code in the SMB header of the response is STATUS_NETWORK_SESSION_EXPIRED, then the client MUST look up the **Client.Connection.SessionTable [UID]**, set **Client.Session.AuthenticationState** to Expired, and attempt to re-authenticate this session. Re-authentication follows the steps as specified in section 3.2.4.2.4, except that the UID sent in the SMB header of the SMB_COM_SESSION_SETUP_ANDX request MUST be set to the UID that represents the expired Session. Also, as described in section 3.2.5.3, the existing Client.Session.SessionKey MUST NOT be modified during re-authentication after a session expiry.

If the authentication fails, then the resulting error code MUST be returned for whichever operation failed with STATUS_NETWORK_SESSION_EXPIRED and the session associated with this UID is removed from the **Client.Connection.SessionTable**. If authentication succeeds, then the client MUST set **Client.Session.AuthenticationState** to Valid and retry the operation that failed with STATUS_NETWORK_SESSION_EXPIRED.

3.2.5.2 Receiving an SMB_COM_NEGOTIATE Response

Processing of an SMB_COM_NEGOTIATE response is handled as specified in [MS-CIFS] section 3.2.5.2 with the following additions:

Storing extended security token

If the capabilities returned in the SMB_COM_NEGOTIATE response include CAP_EXTENDED_SECURITY, then the response MUST take the form defined in section 2.2.4.5.2, and the client MUST set the **Client.Connection.GSSNegotiateToken** to the value returned in the **SecurityBlob** field in the SMB_COM_NEGOTIATE server response.<104>

3.2.5.3 Receiving an SMB_COM_SESSION_SETUP_ANDX Response

The processing of an SMB_COM_SESSION_SETUP_ANDX response is handled as specified in [MS-CIFS] section 3.2.5.3 with the following additions:

Extended Security Authentication

If **Client.Connection.ServerCapabilities** has the CAP_EXTENDED_SECURITY bit set, then the client MUST reject any SMB_COM_SESSION_SETUP_ANDX responses that do not take the form specified in section 2.2.4.6.2. If the **Status** field of the SMB header is not STATUS_SUCCESS and is not STATUS_MORE_PROCESSING_REQUIRED, then the authentication has failed and the error code MUST be propagated back to the application that initiated the connection attempt. Otherwise, if there is no entry in **Client.Connection.SessionTable** for the UID in the response, then one MUST be created with the following additional requirements:

▪ **Client.Session.SessionUID** MUST be set to the UID in the response.

▪ **Client.Session.AuthenticationState** MUST be set to InProgress.

▪ **Client.Session.UserCredentials** MUST be set to the authentication credentials of the user that initiated the authentication attempt.

▪ **Client.Session.SessionKey** MUST be set to zero.

▪ **Client.Session.SessionKeyState** MUST be set to Unavailable.

The client MUST then process the GSS token (the **SecurityBlob** field of the response with its length given in the **SecurityBlobLength** field). If this token is of nonzero length, then the client MUST use the configured GSS authentication protocol to obtain the next GSS token for the authentication exchange. Based on the status code received in the response and the result from the GSS authentication protocol, one of the following actions MUST be taken:

▪ If the GSS authentication protocol indicates an error, then the error MUST be returned to the calling application that initiated the connection. If session state was associated with this authentication, then it MUST be removed from the Client.Connection.SessionTable.

▪ If the **Status** field of the response contains STATUS_SUCCESS and the GSS authentication protocol does not indicate an error, then authentication is complete. The Client.Session.AuthenticationState MUST be set to Valid.

▪ If the **Status** field of the response contains STATUS_MORE_PROCESSING_REQUIRED and the GSS authentication protocol did not indicate an error, then the client MUST create an SMB_COM_SESSION_SETUP_ANDX request (section 2.2.4.6.1) with the following parameters:

  ▪ The client MUST set CAP_EXTENDED_SECURITY in the **Capabilities** field and set SMB_FLAGS2_EXTENDED_SECURITY in the SMB header **Flags2** field.

  ▪ The **SecurityBlob** and **SecurityBlobLength** fields MUST contain the output token generated by the GSS protocol executed earlier.

  ▪ The **SMB_Header.UID** field MUST be set to the value of the **SMB_Header.UID** field of the previous SMB_COM_SESSION_SETUP_ANDX response.

  ▪ This message MUST be sent to the server, and further processing listed in the remainder of this section is not necessary.

NTLM Authentication

If the CAP_EXTENDED_SECURITY bit in **Client.Connection.ServerCapabilities** is not set, then the client processes the response.<105> If the **Status** field of the response does not contain STATUS_SUCCESS, then the client MUST propagate the error to the application that initiated the authentication. The connection MUST remain open for the client to attempt another authentication.

If the **Status** field of the response contains STATUS_SUCCESS, then authentication was successful. The client associates the returned SMB_Header.UID of the response with this user for further requests, as specified in [MS-CIFS]. The Client.Session.AuthenticationState MUST be set to Valid. If the Client.Session.SessionKey is zero, then the client MUST query the authentication package for the 16-byte session key, as specified in [MS-NLMP], and set Client.Session.SessionKey to the returned value. If Client.Session.SessionKey is non-zero, then the client MUST NOT overwrite the existing session key.

Activating Signing

If authentication has completed successfully, then Client.Connection.IsSigningActive is FALSE and the targeted behavior for this connection is signed according to the description in section 3.2.4.2.3. The client MUST determine whether signing is required to be activated.

To determine whether signing is required to be active, the user security context that completed authentication is verified. If the user that authenticated is a guest or is anonymous, then signing MUST NOT be activated. Guest authentication is indicated by bit zero in the **Action** field of the SMB_COM_SESSION_SETUP_ANDX response being set. Anonymous authentication is indicated by the fact that no credentials are provided.

If neither of these conditions are true, then the client MUST activate signing. It MUST query the authentication protocol, either NTLM or via GSS-API, for the session key used in this authentication and store it as **Client.Connection.SigningSessionKey**. If CAP_EXTENDED_SECURITY is set in **Client.Connection.ServerCapabilities**, then it MUST set **Client.Connection.SigningChallengeResponse** to NULL. If CAP_EXTENDED_SECURITY is not set, then it MUST set Client.Connection.SigningChallengeResponse to the challenge response sent in the SMB_COM_SESSION_SETUP_ANDX response.

Once these steps are completed, the client MUST verify the signature of this response. The client follows the steps specified in section 3.1.5.1, by passing in a sequence number of one because this is the first signed packet.

3.2.5.4   Receiving an SMB_COM_TREE_CONNECT_ANDX Response

The processing of an SMB_COM_TREE_CONNECT_ANDX request is handled as specified in [MS-CIFS] section 3.3.5.44 with the following additions:

Requesting Extended Information

The client MUST determine whether or not the server returned an extended response, as specified in section 2.2.4.7. The client does this by determining whether or not the **WordCount** is equal to 0x07. If it is, then the client MUST make the new extended information available to the calling application by using the **SMB_Header.TID** value to set **Client.Connection.TreeConnectTable[TID].MaximalShareAccessRights** and **Client.Connection.TreeConnectTable[TID].MaximalShareAccessRights** to the values that are in the response fields of **SMB_Parameters.Words.MaximalShareAccessRights** and **SMB_Parameters.Words.GuestMaximalShareAccessRights**, respectively.

Session Key Protection

If the response status is STATUS_SUCCESS and the SMB_EXTENDED_SIGNATURE bit is set in the **OptionalSupport** field of the SMB_COM_TREE_CONNECT_ANDX response, then the client MUST hash the session key of the calling user. This protects the key that is used for signing by making it unavailable to the calling applications.

The one-way hash MUST be performed on the user session key that uses the HMAC-MD5 algorithm, as specified in [RFC2104]. The steps are as follows:

1. Take the 16-byte user session key.

   ▪ If this is an LM authentication where the session key is only 8 bytes, then zero extend it to 16 bytes.

   ▪ If the session key is more than 16 bytes, then use only the first 16 bytes.

2. Calculate the one-way hash as follows:

```
CALL hmac_md5( SSKeyHash, 256, session key, session key length, digest )
SET user session key = digest
```

The resulting 16-byte digest is treated as the user's new session key and is returned to any appropriate caller who requests it. SSKeyHash is the well-known constant array that is described in section 2.2.2.5.

After the session key has been hashed, the client MUST place the hash into **Client.Session.SessionKey** and set **Client.Session.SessionKeyState** to Available, which allows applications to query the session key.

If the TREE_CONNECT_ANDX_EXTENDED_SIGNATURE bit is not set, then the **Client.Session.SessionKey** is not changed and **Client.Session.SessionKeyState** MUST be set to Available.

3.2.5.5   Receiving an SMB_COM_NT_CREATE_ANDX Response

The processing of an SMB_COM_NT_CREATE_ANDX response is handled as specified in [MS-CIFS] section 3.2.5.36 with the following additions:

The client MUST determine whether the server returned an extended response, as specified in section 2.2.4.9.2. It does this by checking for a proper **WordCount** value.<106> If the response is not an extended response, then the client MUST process the response as specified in [MS-CIFS] section 3.3.5.50. If the response is an extended response, then the extended information that is specified in section 2.2.4.9.2 MUST also be propagated back to the calling application.

3.2.5.6   Receiving an SMB_COM_OPEN_ANDX Response

The processing of an SMB_COM_OPEN_ANDX response is handled as specified in [MS-CIFS] section 3.2.5.25 with the following additions:

The client MUST determine whether or not the server returned an extended response, as specified in section 2.2.4.1.2. It does this by checking whether or not the **WordCount** is equal to 0x13. If the response is not an extended response, then the client MUST process the response as specified in [MS-CIFS] section 3.3.5.34. If the response is an extended response, then the new information specified in section 2.2.4.1.2 MUST be propagated back to the calling application.

3.2.5.7   Receiving an SMB_COM_READ_ANDX Response

The processing of an SMB_COM_READ_ANDX response is handled as specified in [MS-CIFS] section 3.2.5.26 with the following additions:

The first two bytes of the **SMB_Parameters.Words.Reserved2[]** field, specified in [MS-CIFS] section 2.2.4.42.2, are interpreted as the 16-bit **DataLengthHigh** field specified in section 2.2.4.2.2). The remaining 8 bytes of **Reserved2[]** remain unused and MUST be ignored by the server. The **DataLengthHigh** field MUST contain the two most-significant bytes of a 32-bit length count of bytes read from the server.

3.2.5.8   Receiving an SMB_COM_WRITE_ANDX Response

The processing of an SMB_COM_WRITE_ANDX response is handled as specified in [MS-CIFS] section 3.2.5.27 with the following additions:

The 16-bit **SMB_Parameters.Words.Reserved** field, specified in [MS-CIFS] section 2.2.4.43.2, is now interpreted as a 16-bit **CountHigh** field followed by an 8-bit **Reserved** field. The **CountHigh** field MUST contain the two most-significant bytes of a 32-bit count of bytes written by the server.

3.2.5.9   Receiving any SMB_COM_NT_TRANSACT Response

The processing of any SMB_COM_NT_TRANSACT response is handled as specified in [MS-CIFS] section   3.2.5.40.

3.2.5.9.1   Receiving an NT_TRANSACT_IOCTL Response

The processing of an NT_TRANSACT_IOCTL response is handled as specified in [MS-CIFS] section 3.2.5.40.2 with the following additions.

3.2.5.9.1.1   Receiving an FSCTL_SRV_REQUEST_RESUME_KEY Function Code

If the response indicates that an error occurred, then the client MUST propagate the error to the application that initiated the call.

If the response indicates that the operation is successful, then the client MUST return the copychunk resume key that is received in the Data block of the response to the application that initiated the call. If this is a server-side data copy operation, then the execution MUST continue as specified in section 3.3.5.10.1.2.

3.2.5.9.1.2   Receiving an FSCTL_SRV_COPYCHUNK Function Code

The success or failure code MUST be returned to the calling application. The FSCTL_SRV_COPYCHUNK response (section 2.2.7.2.2) MUST also be returned to the calling application in both success and failure situations. If this is a server-side data copy operation, then the execution MUST continue as specified in section 3.3.5.10.1.3.

3.2.5.9.2   Receiving an NT_TRANSACT_QUERY_QUOTA Response

If the response indicates an error occurred, then the client MUST propagate the error to the application that initiated the call.

If the response indicates the operation is successful, then the client MUST return the information received in the Data block of the response to the application that initiated the call.

3.2.5.9.3   Receiving an NT_TRANSACT_SET_QUOTA Response

The client MUST propagate the success or failure code in the response to the application that initiated the call.

3.2.6 Timer Events

There are no new client timers other than those specified in [MS-CIFS] section 3.2.6.

3.2.7 Other Local Events

There are no new client local events other than those specified in [MS-CIFS] section 3.2.7.

3.3 Server Details

3.3.1 Abstract Data Model

This section specifies a conceptual model of possible data organization that an implementation maintains in order to participate in this protocol. The described organization is provided to explain how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with what is described in this document.

The following elements extend the client abstract data model specified in [MS-CIFS] section 3.3.1.

3.3.1.1 Global

Server.MessageSigningPolicy: This ADM element is extended from the specification in [MS-CIFS] section 3.3.1.1 to include a new possible value:

▪ Declined -- Message signing is disabled unless the other party requires it. If the other party requires message signing, then it MUST be used. Otherwise, message signing MUST NOT be used.

Server.SupportsExtendedSecurity: A flag that indicates whether or not this node supports Generic Security Services (GSS), as specified in [RFC2743], for selecting the authentication protocol.

Server.IsDfsCapable: A Boolean that, if set, indicates that the server supports the Distributed File System.

3.3.1.2 Per Share

Server.Share.SecurityDescriptor: The security descriptor that describes which users can access resources on that share and their corresponding access levels.

Server.Share.SecurityDescriptor is of type SECURITY_DESCRIPTOR, as specified in [MS-DTYP] section 2.4.6. If this value is NULL, then no access limits are enforced.

This structure is shared with [MS-SRVS] in the ShareList as specified in [MS-SRVS] section 3.1.1.1. The following table specifies how local ADM elements map into the shared Share object:

[MS-SRVS] Share	[MS-SMB] Share
shi502_security_descriptor	Server.Share.SecurityDescriptor

Server.Share.IsDfs: A Boolean that, if set, indicates that this share is configured for DFS. For more information, see [MSDFS].

3.3.1.3 Per SMB Connection

Server.Connection.GSSNegotiateToken: A byte array that contains the token received during an extended security negotiation and that is remembered for authentication.

3.3.1.4 Per Pending SMB Command

There is no new state introduced per pending SMB command.

3.3.1.5 Per SMB Session

Server.Session.AuthenticationState: A session can be in one of three states:

- InProgress -- A session setup (an extended SMB_COM_SESSION_SETUP_ANDX exchange, as described in section 3.2.4.2.4.1) is in progress for this session.

- Valid -- The session is valid and a session key and UID are available for this session.

- Expired -- The Kerberos ticket for this session has expired and the session needs to be re-established.

Server.Session.SessionKeyState: The session key state. This can be either Unavailable or Available.

3.3.1.6 Per Tree Connect

There is no new state introduced per tree connect.

3.3.1.7 Per Unique Open

Server.Open.GrantedAccess: The access level granted on this Open.

3.3.2 Timers

3.3.2.1 Authentication Expiration Timer

The Authentication Expiration Timer, a re-authentication timer, is used to mark an authentication as expired when its authentication-specific expiration time is reached. NTLM authentication has no expiration time, so authentications done with NTLM do not expire. The default expiration for Kerberos is implementation specific.<107>

3.3.3 Initialization

The following values MUST be initialized at system startup:

- **Server.MessageSigningPolicy** and **Server.SupportsExtendedSecurity** MUST be set based on system policy.<108> The value of this is not constrained by the values of any other policies.

When an SMB connection is established, the following values MUST be initialized.

- **Server.Connection.GSSNegotiateToken** MUST be empty.

When an SMB session is established on an SMB connection, the following value MUST be initialized:

- **Server.Session.AuthenticationState** MUST be set to InProgress.

- **Server.Session.SessionKeyState** MUST be set to Unavailable.

- **Server.IsDFSCapable** MUST be set to FALSE.

- **Server.Share.IsDfs** MUST be set to FALSE.

All other values are initialized as specified in [MS-CIFS] section 3.3.3.

3.3.4 Higher-Layer Triggered Events

3.3.4.1 Sending Any Message

This interface is used internally by the server to send a message to the client. It is not exposed to external callers.

No new global details are presented to a server that sends any message beyond what is specified in [MS-CIFS] section 3.3.4.1.

3.3.4.1.1 Sending Any Error Response Message

In response to an error in the processing of any SMB request, the server SHOULD<109> follow the format as specified in [MS-CIFS] section 3.3.4.1.2.

3.3.4.2 Server Application Queries a User Session Key

The application MUST provide:

- The security context of the user whose session is being sought.

- Either a valid ClientName or a valid **Open**.

The server MUST locate an appropriate SMB connection that uses either the application-supplied **ServerName** to look in the **Server.ConnectionTable[ClientName]** or the application-supplied **Open.Connection**. If a valid **Connection** is found, then the server MUST scan for an SMB session in the **Server.Connection.SessionTable** that matches the security context of the user. If no entry is found, then the application request MUST be failed with STATUS_INVALID_PARAMETER. If a Session is found but **Server.Session.SessionKeyState** is Unavailable, then the request MUST be failed with STATUS_ACCESS_DENIED. If **Server.Session.SessionKeyState** is Available, then the 16-byte **Server.Session.SessionKey** MUST be returned to the calling application.

3.3.4.3 DFS Server Notifies SMB Server That DFS Is Active

In response to this event, the SMB server MUST set the global state variable **Server.IsDfsCapable** to TRUE. If the DFS server is running on this computer, it MUST notify the SMB server that the DFS capability is available via this event.

3.3.4.4 DFS Server Notifies SMB Server That a Share Is a DFS Share

In response to this event, the SMB server MUST set the **Server.Share.IsDfs** to TRUE. When a DFS server running on this computer claims a share as a DFS share, it MUST notify the SMB server via this event.

3.3.4.5 DFS Server Notifies SMB Server That a Share Is Not a DFS Share

In response to this event, the SMB server MUST clear the Server.ShareIsDfs attribute of the share specified in section 3.3.1.2

3.3.5 Processing Events and Sequencing Rules

3.3.5.1 Receiving Any Message

The following global details are presented to a server that receives any message in addition to what is specified in [MS-CIFS] section 3.3.5.1.

Signing

If a message is received and **Server.Connection.IsSigningActive** is TRUE, then the signature MUST be verified, as specified in section 3.1.5.1.

The server is responsible for providing the expected sequence number for signature validation. The sequence number for the next incoming request is stored in **Server.Connection.ServerNextReceiveSequenceNumber**. The server MUST remember the appropriate sequence number for the response to this request and does so by inserting it into the **Server.Connection.ServerSendSequenceNumber** table with the **PID** and **MID** that identifies the request/response pair. (**PID** and **MID** are specified in [MS-CIFS] section 2.2.1.6.)

If the signature on the received packet is incorrect, then the server SHOULD<110> terminate the connection. If the signature on the current message is correct, then the server MUST take the following steps.

```
IF request command EQUALS SMB_COM_NT_CANCEL THEN
    INCREMENT ServerNextReceiveSequenceNumber
ELSE IF request has no response THEN
    INCREMENT ServerNextReceiveSequenceNumber BY 2
ELSE
    SET ServerSendSequenceNumber[PID,MID] TO
        ServerNextReceiveSequenceNumber + 1
    INCREMENT ServerNextReceiveSequenceNumber BY 2
END IF
```

Session Validation and Re-authentication

If the **SMB_Header.UID** of the request is zero, then the server does not need to check for the expiry because a session is not being used for this request.

If the **SMB_Header.UID** of the request is not zero, then the server MUST check the state of the session.

- If Connection.SessionTable[UID].AuthenticationState is equal to Expired,

- If a session renewal is indicated and is successful (SMB_COM_SESSION_SETUP_ANDX), then the server MUST allow all operations as though the session had never expired. For details on how the client handles a session expiration, see section 3.2.5.1.

- Otherwise, the server MUST fail this operation with STATUS_NETWORK_SESSION_EXPIRED.

- If **Server.Connection.SessionTable[UID].AuthenticationState** is InProgress, then the server MUST fail this operation with STATUS_ACCESS_DENIED, unless it is a SMB_COM_SESSION_SETUP_ANDX for session establishment.

3.3.5.1.1 Scanning a Path for a Previous Version Token

If a request is a path-based operation (for example, SMB_COM_NT_CREATE_ANDX) and has SMB_FLAGS2_REPARSE_PATH set in the **Flag2** field of the SMB header, then the server MUST perform a parse of the path by checking for previous version tokens (section 2.2.1.1.1). If the flag is not set, then the server MAY<111> parse the path anyway.

If a previous version token is found in the pathname, but the file or directory does not exist for the given snapshot, then the server MUST fail the operation with STATUS_OBJECT_NAME_NOT_FOUND. If the file or directory does exist, then processing continues as normal, except that the execution is against the previous version selected.

3.3.5.2 Receiving an SMB_COM_NEGOTIATE Request

The processing of an SMB_COM_NEGOTIATE request is handled as specified in [MS-CIFS] section 3.3.5.41, with the following additions:

New Capabilities

The new capabilities flags specified in section 2.2.4.5.1 MUST also be considered when setting the **SMB_Parameters.Words.Capabilities** field of the response based on the **Server.Capabilities** attribute.

Generating Extended Security Token

If the client indicated support for extended security by setting SMB_FLAGS2_EXTENDED_SECURITY in the Flags2 field of the SMB header of the SMB_COM_NEGOTIATE request, then the server SHOULD set CAP_EXTENDED_SECURITY in the SMB_COM_NEGOTIATE response if it supports extended security. The response MUST take the form specified in section 2.2.4.1.2.

The server SHOULD set the **SecurityBlob** of the SMB_COM_NEGOTIATE response to the first GSS token (or fragment thereof) produced by the GSS authentication protocol it is configured to use (GSS tokens are as specified in [RFC2743]). Otherwise, it leaves it empty. This token is also stored in **Server.Connection.GSSNegotiateToken**.

The server MUST initialize its GSS mechanism with the Integrity, Confidentiality, and Delegate options and use the Server-Initiated variation, as specified in [RFC4178]. The SMB_COM_NEGOTIATE response packet is sent to the client.<112>

3.3.5.3 Receiving an SMB_COM_SESSION_SETUP_ANDX Request

The processing of an SMB_COM_SESSION_SETUP_ANDX request is handled as specified in [MS-CIFS] section 3.3.5.42 with the following additions:

Storing Client Capabilities

If **Server.Connection.ClientCapabilities** is equal to zero, then the server MUST set **Server.Connection.ClientCapabilities** to the **Capabilities** field that is received in the SMB_COM_SESSION_SETUP_ANDX request. If **Server.Connection.ClientCapabilities** has already been determined and is nonzero, then the server MUST ignore the capabilities value on subsequent requests.

Determine Reauth or Continuation of Previous Auth

If the **SMB_Header.UID** is not zero, then the server MUST look up the authentication state for this session and take the following actions based on this state.

- If **Server.Connection.SessionTable[UID].AuthenticationState** is InProgress, then this is a continuation of an authentication in progress. This state indicates that the authentication required multiple roundtrips, and that authentication continues.

- If **Server.Connection.SessionTable[UID].AuthenticationState** is Complete or Expired, then this is the re-authentication of a user. The server MUST set AuthenticationState to InProgress and begin a new authentication for this session. The server MUST prevent any further operations from executing on this session until authentication is complete, and fail them with STATUS_NETWORK_SESSION_EXPIRED.

- If there is no session for the provided UID, then the request MUST be failed with STATUS_SMB_BAD_UID.

Extended Security

If CAP_EXTENDED_SECURITY is set in **Server.Connection.ClientCapabilities**, then the server MUST handle the authentication as defined in this section. Otherwise, it SHOULD continue to the following NTLM authentication section.

The server MUST extract the GSS token, which is the **SecurityBlob** contained in the request, with a length of **SecurityBlobLength**. The server MUST use the configured GSS authentication protocol to obtain the next GSS output token for the authentication protocol exchange. Note that this token can be 0 bytes in length.

If the GSS mechanism indicates an error that is not STATUS_MORE_PROCESSING_REQUIRED, then the server MUST fail the client request, and return only an SMB header and propagate the failure code. If a **UID** was present in this request, then its associated session MUST be removed from the **Server.Connection.SessionTable**. The authentication has failed and no further processing is done on this request. This error response is sent to the client.

If the GSS mechanism indicates success, then the server MUST create an SMB_COM_SESSION_SETUP_ANDX response (section 2.2.4.6.2). The **SecurityBlob** MUST be set to the output token from the GSS mechanism, and **SecurityBlobLength** is set to the length of the output token. SMB_FLAGS2_EXTENDED_SECURITY is set in the **Flags2** field of the SMB header of the response. If the request did not specify a **UID** in the SMB header of the request, then a **UID** MUST be generated to represent this user's authentication and its value MUST be placed in the UID field of the SMB header of the response.

If the GSS mechanism indicates that the current output token is the last output token of the authentication exchange based on the return code, as specified in [RFC2743], then the **Status** field in the SMB header of the response MUST be set to STATUS_SUCCESS, and **Server.Connection.SessionTable[UID].AuthenticationState** MUST be set to Completed. Otherwise, the **Status** field in the SMB header of the response MUST be set to STATUS_MORE_PROCESSING_REQUIRED, and **Server.Connection.SessionTable[UID].AuthenticationState** MUST be set to InProgress.The processing for this request continues in the Signing Initialization subsection later in this section.

Activating Signing

If **Server.Connection.IsSigningActive** is FALSE, and the response of the SMB_COM_SESSION_SETUP_ANDX operation contains STATUS_SUCCESS, then the server MUST determine whether or not signing can be activated.

If bit zero of the **Action** field of the SMB_COM_SESSION_SETUP_ANDX response is set, then signing MUST NOT be activated. If the value of this field is one, then the user attempted to log in as a user other than Guest, but could not be authenticated for that account. Using a fallback mechanism on the server, the user is now logged in as Guest.

Otherwise, **Server.Connection.IsSigningActive** MUST be set to TRUE. The server MUST query the authentication protocol, either using NTLM or via GSS API, for the session key used in this authentication, and store it as **Server.Connection.SigningSessionKey**. If CAP_EXTENDED_SECURITY is set in **Server.Connection.ClientCapabilities**, then it MUST set **Server.Connection.SigningChallengeResponse** to NULL. If that capability is not set, then it MUST set **Server.Connection.SigningChallengeResponse** to the challenge response received in the SMB_COM_SESSION_SETUP_ANDX request.

Once these steps are performed, the server MUST sign the SMB_COM_SESSION_SETUP_ANDX response. The server follows the steps as specified in section 3.1.5 by passing in an index number of one.

Acquire Session Key

If authentication is successful, the server MUST query the session key from the authentication package (as specified in [MS-NLMP] for implicit NTLM and in [RFC4178] for extended security). The server MUST then place this value into **Server.Session.SessionKey**, and MUST set **Server.Session.SessionKeyState** to Unavailable.

Authentication Expiry

If the authentication (either NTLM or GSS processing) returned an expiration time, such as a Kerberos ticket time-out, then the server MUST set up a Timer to mark the **Server.Session.AuthenticationState** as Expired when the time-out occurs.

3.3.5.4   Receiving an SMB_COM_TREE_CONNECT_ANDX Request

The processing of an SMB_COM_TREE_CONNECT_ANDX request is handled as specified in [MS-CIFS] section 3.3.5.44 with the following additions:

Requesting Extended Information

If the TREE_CONNECT_ANDX_EXTENDED_RESPONSE is set in the **Flags** field of the SMB_COM_TREE_CONNECT_ANDX request, then the server MUST respond with the structure specified in section 2.2.4.7.2.

The server MUST populate the **SMB_Parameters.Words.OptionalSupport** field of the response with a value of **Server.Share.OptionalSupport**.

The server MUST calculate the maximal share access rights for the user that requests the tree connect using the following algorithm.

```
MaxRights = 0x00000000
IF Server.Share.SecurityDescriptor == NULL
  MaxRights = 0xFFFFFFFF
ELSE
  FOR EACH AccessBit value defined in section 2.2.1.4
    Compute access for the user, using Server.Share.SecurityDescriptor and
     Server.Session.SecurityContext, as described in [MS-DTYP] section 2.5.2.1.
    IF access was granted
      MaxRights = MaxRights | AccessBit;
    END IF
```

```
      END FOR
   END IF
```

The computed MaxRights ACCESS_MASK MUST be placed in the
SMB_Parameters.Words.MaximalShareAccessRights of the response. If no access is granted
for the client on this share, the server MUST fail the request with STATUS_ACCESS_DENIED.

Using the same algorithm, the **SMB_Parameters.Words.GuestMaximalAccessRights** field of the
response MUST be set to the calculated highest access rights the guest account has on this share.
Instead of using **Server.Session.SecurityContext**, the server MUST use the guest account's
security context. If the system does not support the guest account, then it MUST set
GuestMaximalAccessRights to zero.

Session Key Protection

If the client has set the TREE_CONNECT_ANDX_EXTENDED_SIGNATURE bit in the **Flags** field of the
SMB_COM_TREE_CONNECT_ANDX request, then the server MUST hash the session key of the
calling user. This protects the key used for signing by making it unavailable to server-side
applications.

The one-way hash MUST be performed on the user session key by using the HMAC-MD5 algorithm,
as specified in [RFC2104]. The steps are as follows:

1. Take the 16-byte user session key.

   ▪ If this is an LM authentication where the session key is only 8 bytes, then zero extend it to 16
     bytes.

   ▪ If the session key is more than 16 bytes, then use only the first 16 bytes.

2. Calculate the one-way hash as follows:

   ```
   CALL hmac_md5( SSKeyHash, 256, session key, session key length, digest )
   SET user session key = digest
   ```

   The resulting 16-byte digest is treated as the user's new session key and returned to any
   appropriate caller who requests it. SSKeyHash is the well-known constant array that is described
   in section 2.2.2.5.

After the session key has been hashed, the server MUST place the hash into
Server.Session.SessionKey and set **Server.Session.SessionKeyState** to Available, which allows
applications to query the session key. If the TREE_CONNECT_ANDX_EXTENDED_SIGNATURE bit is
not set, then the **Server.Session.SessionKey** is not changed and
Server.Session.SessionKeyState MUST be set to Available.

3.3.5.5   Receiving an SMB_COM_NT_CREATE_ANDX Request

The processing of an SMB_COM_NT_CREATE_ANDX request is handled as specified in [MS-CIFS]
section 3.3.5.50 with the following additions:

On a successful create or open, if the NT_CREATE_REQUEST_EXTENDED_RESPONSE flag was set in
the **Flags** field of the request, then the server SHOULD<113> send an extended response (section
2.2.4.9.2).

If the server sends the new response, then it MUST construct a response as specified in section 2.2.4.9.2. with the addition of the following rules:

▪ The server MUST query the file attributes and use them to set the **FileStatusFlags** in the response.<114>

▪ If the underlying object store of the share in which the file is opened or created does not support streams, then the server MUST set the NO_SUBSTREAMS bit in the **FileStatusFlags** field.<115>

▪ The server MAY<116> fill in the **VolumeGUID** and **FileId** fields if it wants the client to leverage the new capabilities. Otherwise, they MUST be set to zero.

▪ The server MUST query the underlying object store for the granted access rights on the returned **Server.Open** and store them in the **MaximalAccessRights** field of the response. If the file has no security applied, the field MUST be set to 0xFFFFFFFF. Likewise, it MUST query the granted access a Guest user has on this file and place it in the **GuestMaximalAccessRights** field of the response. The server MUST use **Open.Session.UserSecurityContext** and the impersonation level given in the **ImpersonationLevel** field of the request to impersonate the client. If the impersonation level is SECURITY_ANONYMOUS, then the server MUST fail the request with STATUS_BAD_IMPERSONATION_LEVEL.<117>

▪ The server SHOULD<118> ignore all **CreateOptions** on a named pipe.

If **Server.IsDfsCapable** is TRUE and **Server.Share.IsDfs** is True, then server MUST invoke the interface defined in [MS-DFSC] section 3.2.5.6 to normalize the pathname by supplying the target path name. The normalized name returned from the DFS subsystem MUST be used for subsequent processing. If normalization fails, the server MUST fail the create request with the error code returned by the DFS normalization routine. Otherwise, the server MUST continue processing.

3.3.5.6   Receiving an SMB_COM_OPEN_ANDX Request

The processing of an SMB_COM_OPEN_ANDX request is handled as specified in [MS-CIFS] section 3.3.5.34 with the following additions:

On a successful open, if the SMB_OPEN_EXTENDED_RESPONSE flag was set in the **Flags** field of the request, then the server MAY<119> send an extended response (section 2.2.4.1.2) if it wants the client to leverage the new capabilities.

If the server chooses to send the new response, then it MUST construct a response as detailed in section 2.2.4.1.2. The server MUST query the underlying object store for the granted access rights on the returned **Server.Open** and store them in the **MaximalAccessRights** field of the response. If no access is granted for the client on this share, the server MUST fail the request with STATUS_ACCESS_DENIED. If the file has no security applied, then the field MUST be set to 0xFFFFFFFF. Likewise, it MUST query the granted access a Guest user has on this file and place it in the **GuestMaximalAccessRights** field of the response. The server MUST use Open.Session.UserSecurityContext and the impersonation level given in the **ImpersonationLevel** field of the request to impersonate the client. If the impersonation level is SECURITY_ANONYMOUS, then the server MUST fail the request with STATUS_BAD_IMPERSONATION_LEVEL.<120>

3.3.5.7   Receiving an SMB_COM_READ_ANDX Request

The processing of an SMB_COM_READ_ANDX request is handled as specified in [MS-CIFS] section 3.3.5.35 with the following additions:

If the read operation is on a named pipe, then the **Timeout_or_MaxCountHigh** field MUST be interpreted as the 32-bit **Timeout** field, as specified in section 2.2.4.1.2.

If the read operation is on a file, then the **Timeout_or_MaxCountHigh** field MUST be interpreted as the 16-bit **MaxCountHigh** field followed by a 16-bit **Reserved** field, as specified in section 2.2.4.1.1. The value in **MaxCountHigh** MUST be treated as the two most significant bytes of the count of bytes to read and is combined with the value of **MaxCountOfBytesToReturn** to create a 32-bit count of bytes to read (as specified in section 3.2.4.4). If **MaxCountHigh** is set to 0xFFFF, then the value MUST be ignored, and only the length received in **MaxCountOfBytesToReturn** is used.

It is acceptable to return fewer bytes than requested by the client, with the restriction that reads from named pipes or devices MUST return at least **MinCountOfBytesToReturn** bytes. If the count of bytes read is greater than or equal to 0x00010000 (64K), then the server MUST set the two least significant bytes of the count in the **DataLength** field in the response, and the two most significant bytes of the count in the **DataLengthHigh** field (specified in section 2.2.4.2.2).<121>

3.3.5.8   Receiving an SMB_COM_WRITE_ANDX Request

The processing of an SMB_COM_WRITE_ANDX request is handled as specified in [MS-CIFS] section 3.3.5.36 with the following additions:

If CAP_LARGE_WRITEX is set in **Server.Connection.ClientCapabilities**, then it is possible that the count of bytes to be written is larger than the server's **MaxBufferSize**. The count of bytes to be written is specified in the **DataLength** and **DataLengthHigh** fields sent in the request, as specified in section 2.2.4.3.1.

If the server successfully writes data to the underlying object store, then the count of bytes written MUST be set in the **Count** and **CountHigh** fields of the response, as specified in section 2.2.4.3.2.

3.3.5.9   Receiving any SMB_COM_TRANSACTION2 subcommand

The processing of any SMB_COM_TRANSACTION2 subcommand request is handled as specified in [MS-CIFS] section 3.3.5.57 with the following additions:

3.3.5.9.1   Receiving any Information Level

If the server receives client request with a pass-through Information Level (section 2.2.2.3.5) and the server supports the CAP_INFOLEVEL_PASSTHRU capability in **Server.Capabilities**, then the server MUST decrement the Information Level value by SMB_INFO_PASSTHROUGH by treating the value as little-endian, and pass that value to the underlying object store. If the Information Level includes any request data, then the data MUST also be passed to the underlying object store.<122>

If the server does not support pass-through Information Levels, then it MUST fail this request with STATUS_INVALID_PARAMETER.

The returned status and response data, if any, are sent to the client in a Trans2 subcommand response message that corresponds to the same subcommand that initiated the request.

3.3.5.9.2   Receiving a TRANS2_FIND_FIRST2 Request

New Information Levels

The server SHOULD allow for the new Information Levels, as specified in section 2.2.2.3.1. If the server does not support the new Information Levels, then it MUST fail the operation with STATUS_NOT_SUPPORTED.<123>

Enumerating Previous Versions

If a scan for previous version tokens (section 3.3.5.1.1) reveals that the **FileName** of the TRANS_FIND_FIRST2 request contains the search pattern @GMT-* and the requested Information Level is SMB_FIND_FILE_BOTH_DIRECTORY_INFO, then the server MAY<124> choose to return an enumeration of previous versions that are valid for the share. It does this by manufacturing a file entry for each previous version, as defined in section 2.2.8.1.1. If the server chooses not to do this, then the enumeration MUST be processed as a normal TRANS2_FIND_FIRST2 operation.

3.3.5.9.3   Receiving a TRANS2_FIND_NEXT2 Request

New Information Levels

If the query is started using one of the new Information Levels, as specified in section 2.2.2.3.1, then the same Information Level structure MUST be used for the return of subsequent entries in the enumeration continuation.

Enumerating Previous Versions

Likewise, a query for previous version information that is started MUST be continued at the client's request with further entries generated, as defined in section 3.3.5.9.1.

3.3.5.9.4   Receiving a TRANS2_QUERY_FILE_INFORMATION Request

Pass-through Information Levels

If the client requests a pass-through Information Level, then the processing follows as specified in section 3.3.5.9.1.

3.3.5.9.5   Receiving a TRANS2_QUERY_PATH_INFORMATION Request

Pass-through Information Levels

If the client requests a pass-through Information Level, then the processing follows as specified in section 3.3.5.9.1.

3.3.5.9.6   Receiving a TRANS2_SET_FILE_INFORMATION Request

Pass-through Information Levels

If the client requests a pass-through Information Level, then the processing follows as specified in section 3.3.5.9.1.<125>

3.3.5.9.7   Receiving a TRANS2_SET_PATH_INFORMATION Request

Pass-through Information Levels

If the client requests a pass-through Information Level, then the processing follows as specified in section 3.3.5.9.1.<126>

3.3.5.9.8   Receiving a TRANS2_QUERY_FS_INFORMATION Request

Pass-through Information Levels

If the client requests a pass-through Information Level, then the processing follows as specified in section 3.3.5.9.1.

3.3.5.9.9   Receiving a TRANS2_SET_FS_INFORMATION Request

If the client requests a pass-through Information Level, then processing follows as specified in section 3.3.5.9.1.

There is no way to know if a server file system supports a given Information Level. From a protocol perspective, if a client issues a request and it fails with STATUS_NOT_SUPPORTED, then it MUST be inferred that the server file system does not support the request.

3.3.5.10   Receiving any SMB_COM_NT_TRANSACT Subcommand

The processing of any SMB_COM_NT_TRANSACT subcommand request is handled as specified in [MS-CIFS] section 3.3.5.58 with the following additions specified in section 3.3.5.10.1:

3.3.5.10.1   Receiving an NT_TRANSACT_IOCTL Request

The new SMB-specific FSCTL codes are not directly passed to the underlying object store. Instead, processing is as specified in section 3.3.5.10.1.1.

3.3.5.10.1.1   Receiving an FSCTL_SRV_ENUMERATE_SNAPSHOTS Function Code

This is a request to enumerate the available previous versions for a share. The server MUST return an enumeration of available previous versions, as specified in section 2.2.7.2.2. The **NumberOfSnapshots** MUST contain the total number of previous versions that are available for the volume and **NumberOfSnapshotsReturned** contains the number of entries that are returned in this enumeration. If **MaxDataCount** is not large enough to hold all of the entries, then the server SHOULD return zero entries. The value returned in **SnapShotArraySize** MUST be the size required to receive all of the available previous versions. If the **MaxDataCount** of the request is smaller than the size of an FSCTL_ENUMERATE_SNAPSHOTS response, then the server MUST fail the request with STATUS_INVALID_PARAMETER.

If the server does not support this operation, then it SHOULD fail the request with STATUS_NOT_SUPPORTED.

3.3.5.10.1.2   Receiving an FSCTL_SRV_REQUEST_RESUME_KEY Function Code

This is a request for an opaque copychunk resume key for use in an FSCTL_SRV_COPYCHUNK operation. The server MUST generate a 24-byte value that is used to uniquely identify the open of the file against which this operation is executed.

If this operation is successful, then the server MUST construct an FSCTL_SRV_CHOPYCHUNK response as specified in section 2.2.7.2.2, with the following additional requirements:

The **CopychunkResumeKey** field MUST be the server-generated value.

If the server does not support this operation, then it MUST fail the request with STATUS_NOT_SUPPORTED.

3.3.5.10.1.3   Receiving an FSCTL_SRV_COPYCHUNK Request

This is a request for a server-side data copy as specified in section 2.2.7.2.1. The server MUST identify the source file based on the copychunk resume key field of the FSCTL_SRV_COPYCHUNK request. This copychunk resume key is a value that was returned by the server from an FSCTL_SRV_REQUEST_RESUME_KEY operation. If the copychunk resume key is not valid or does not represent an open file, then the server MUST fail the operation with

STATUS_OBJECT_NAME_NOT_FOUND. If the file represented by the resume key is not opened for read-data access, then the server MUST fail the operation with STATUS_ACCESS_DENIED. Likewise, the target file MUST be specified by the Fid in the SMB_COM_NT_TRANSACTION request. If the target file is not opened for write-data access, then the server MUST fail the operation with STATUS_ACCESS_DENIED.

The server MUST validate that the amount of data to be written is within the server's configured bounds. If the number of chunks is too high, if the maximum write size of any individual chunk is too large, or if the total write size of all chunks in the request is too large, then the server MUST fail the operation with STATUS_INVALID_PARAMETER and return a response as specified in section 2.2.7.2.2. A **ChunksWritten** request MUST contain the maximum number of chunks that can be written in a single request. A **ChunkBytesWritten** request MUST be the maximum number of bytes that can be written in a single chunk. A **TotalBytesWritten** request MUST be the total amount of data that can be written in an FSCTL_SRV_COPYCHUNK operation.

If both the source and the target file are appropriate, then the server MUST iterate through the data ranges specified in the request by reading data from the source offset of the source file and writing it to the target offset of the target file. If an error is encountered, then the server MUST stop and set the response parameters, as specified in section 2.2.7.2.2, to indicate how much data was successfully written.

The server MAY restrict the amount of processing that can be done for a single FSCTL_SRV_COPYCHUNK operation by the amount of time it has to process. If the timeout restriction is not enforced, then the server MUST process all of the chunks that are described by the request, as specified in section 2.2.7.2.2.<127>

If the server does not support this operation, then it MUST fail the request with STATUS_NOT_SUPPORTED.

3.3.5.10.2   Receiving an NT_TRANS_QUERY_QUOTA Request

The server MUST attempt to enumerate quota information from the underlying object store on which the file or directory indicated by the **Server.Open** identified by the **SMB_Parameters.Words.Setup.FID** field of the request resides.<128>

The server MUST return as much of the available quota information that is able to fit in the maximum response buffer size denoted by **MaxDataCount**. If the entire quota information cannot fit in the response buffer, then the server MUST return a status of STATUS_BUFFER_TOO_SMALL. Otherwise, the server MUST return STATUS_SUCCESS. The format of the request determines which entries need to be returned, as specified in section 2.2.7.3.1. The server MUST place the quota information in the response, as specified in section 2.2.7.3.2, and send the response back to the client.

3.3.5.10.3   Receiving an NT_TRANS_SET_QUOTA Request

The server MUST attempt to apply the provided quota information to the underlying object store on which the file or directory indicated by the Fid resides. If the underlying object store does not support quotas, then the server SHOULD<129> return STATUS_NOT_SUPPORTED.

The server MUST apply the quota information provided in the **NT_Trans_Data** block of the request (see section 2.2.7.4.1).

The resulting success or error received from the underlying object store is returned in the response, as specified in section 2.2.7.4.2.

3.3.6 Timer Events

3.3.6.1 Authentication Expiration Timer Event

When the Authentication Expiration Timer expires, the server scans the active authentication tokens and marks the Authentication state for those whose expiry time has passed. It MUST set **Server.Session.AuthenticationState**[**UID**] = *Expired*.

3.3.7 Other Local Events

There are no new server local events other than those specified in [MS-CIFS] section 3.3.7.

4 Protocol Examples

The following sections describe common scenarios that indicate normal traffic flow on the wire in order to illustrate the extensions to **CIFS** that are specified in this document.

4.1 Extended Security Authentication

The following diagram depicts the protocol message sequence for a multi-phase extended security exchange and previous versions enumeration and access on the share root folder.
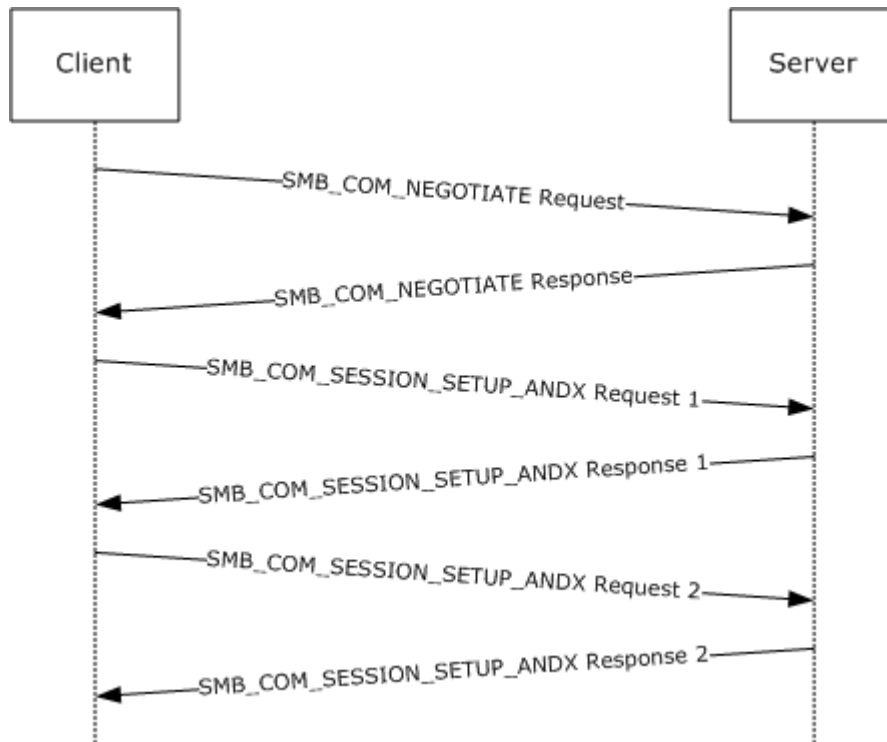

Figure 4: User authentication and session establishment sequence

Descriptions of the fields in this example are specified in [MS-CIFS] and section 2.2.4.1. Fields that are shown and highlighted in bold text are relevant to this extension. It is assumed that the client has successfully established a network connection with the server.

The client initiates the first message with an SMB_COM_NEGOTIATE request, as specified in [MS-CIFS]. The client specifies extended security negotiation in the header **Flags2** field. It also includes NT LM 0.12 in the dialect strings list. The server constructs an extended SMB_COM_NEGOTIATE response packet that is denoted by the *WordCount* field. The server returns dialect index, its capabilities, GUID value, and the initial security binary large object (BLOB) obtained, as specified in [RFC2743] and defined in the preceding figure.

FRAME 1. Client negotiate request

```
Client -> Server: Command = SMB_COM_NEGOTIATE
Flags2 Summary = 51207 (0xC807)
  1100 1000 0000 0111
  .... 1...  ....  ....  = Extended security negotiation is supported
```

Release: Tuesday, April 6, 2010

```
Dialect Strings
PC NETWORK PROGRAM 1.0
LANMAN1.0
Windows for Workgroups 3.1a
LM1.2X002
LANMAN2.1
NT LM 0.12
```

FRAME 2. Server negotiate response

```
Server -> Client: Command = SMB_COM_NEGOTIATE
NT status code = 0x0, STATUS_SUCCESS
Word count = 17
Protocol Index = 5  (NT LM 0.12)
Capabilities = 2147607549 (0x8001F3FD)
 1000 0000 0000 0001 1111 0011 1111 1101
 .... .... .... .... ..1. .... .... ....  = Supports Pass-Thru levels
 1... .... .... .... .... .... .... ....  = Supports extended security
Server GUID = 01 B3 1E 23 07 2A A4 4D A1 9F B6 69 F0 45 71 90
Security Blob in payload
```

The client uses the initial security BLOB that is returned by the server along with any user credential information in order to obtain its security BLOB, as specified in [RFC2743] and defined in section 3.2.4.2.4. The resulting security BLOB is sent to the server as part of the SMB_COM_SESSION_SETUP_ANDX extended request. The client also sends its capabilities and zero **UID** to mark the start of a new session setup exchange. The server verifies that the client requests extended security by checking the **Flags2** and **Capabilities** fields in the request, accepts as input the client security BLOB, and processes it, as specified in [RFC2743]. In this case, the security package requires more processing and returns a second security BLOB to be returned to the client. Also, the server allocates a new UID and associates it with this session setup exchange.

Note Extended security may require multiple request and response exchanges between client and server to complete. The **UID** is defined by the server on first response to an extended session setup and is used for the lifetime of the session.

FRAME 3. Client request for extended session setup

```
Client -> Server: Command = SMB_COM_SESSION_SETUP_ANDX
Header: Tid = 0x0000 Mid = 0x0070 Uid = 0x0000
Flags2 Summary = 51207 (0xC807)
 1100 1000 0000 0111
 .... 1... .... ....  = Supports extended security
Word count = 12
Capabilities = 0xA0000000
1010 0000 0000 0000 0000 0000 0000 0000
..1. .... .... .... .... .... .... ....  = Supports dynamic reauth
1... .... .... .... .... .... .... ....  = Requests extended security
Security Blob Length = 74 (0x4A)
Security Blob in payload
```

FRAME 4. Server response with session setup continuation

```
Server -> Client: Command = SMB_COM_SESSION_SETUP_ANDX
NT status code = 0xC0000016, STATUS_MORE_PROCESSING_REQUIRED
Header: Tid = 0x0000 Mid = 0x0070 Uid = 0x0802
Flags2 Summary = 51207 (0xC807)
  1100 1000 0000 0111
   .... 1... .... ....  = Extended security negotiation is supported
Security Blob Length = 349 (0x15D)
          Security Blob in payload
```

The client accepts as input the server security BLOB and processes it, as specified in [RFC2743], and its output is returned to the server along with the UID. The server uses the **UID** value to associate this request with the pending session establishment. The server processes this request, as specified in [RFC4178], and receives a success result. At this point, the SMB_SESSION_SETUP_ANDX exchange is complete because the status code is not equal to STATUS_MORE_PROCESSING_REQUIRED. The final security BLOB is returned with the success indication.

FRAME 5. Client session setup request continuation

```
Client -> Server: Command = SMB_COM_SESSION_SETUP_ANDX
Header: Tid = 0x0000 Mid = 0x0080 Uid = 0x0802
Flags2 Summary = 51207 (0xC807)
 1100 1000 0000 0111
   .... 1... .... ....  = Extended security negotiation is supported
Word count = 12
Security Blob Length = 226 (0xE2)
          Security Blob in payload
```

FRAME 6. Server response with session setup completion

```
Server -> Client: Command = SMB_COM_SESSION_SETUP_ANDX
NT status code = 0x0, STATUS_SUCCESS
Header: Tid = 0x0000 Mid = 0x0080 Uid = 0x0802
Security Blob Length = 9 (0x9)
Security Blob in payload
```

At this point, the client has been successfully authenticated.

4.2   Previous File Version Enumeration

The following example shows how the client accesses a previous version of the share root folder. It is assumed that the client has already authenticated, established a tree connect to the target share, and opened a handle to the root directory, as specified in [MS-CIFS]. Thus, Frame 1 is not truly the first frame for the connection, but it is referred to as the starting point for this operation.
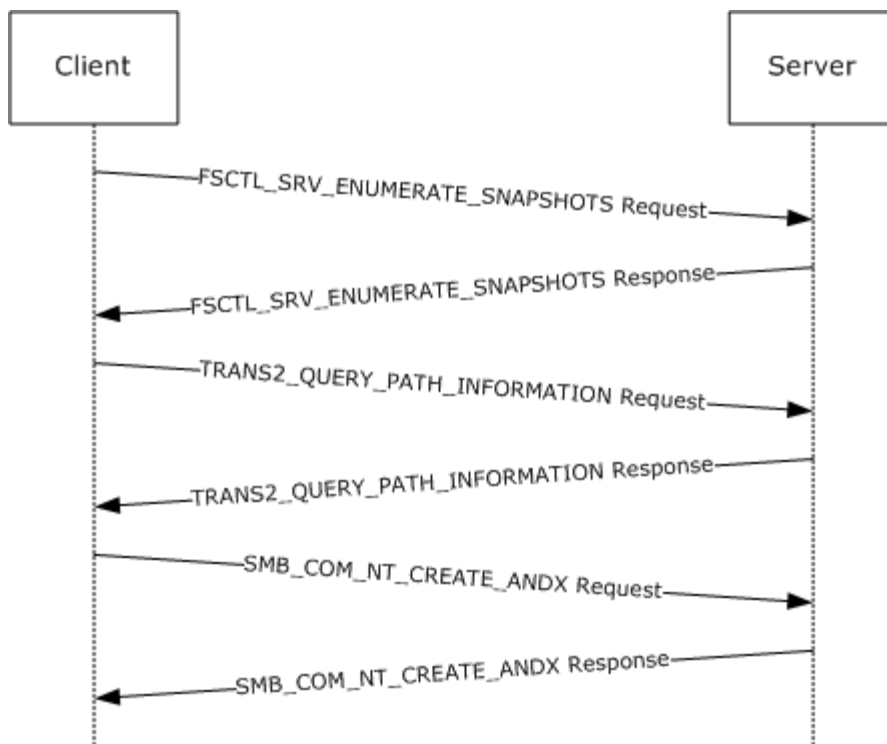
Figure 5: Previous file version enumeration sequence

The first step is to enumerate the list of available snapshots on the server by using the FSCTL_SRV_ENUMERATE_SNAPSHOT command. The client requests the list of snapshots that are available on the server by using the root handle Fid. The server returns the list of snapshots in the format that is defined in the preceding figure. In this example, the server has one snapshot total for the root folder, the payload contains one snapshot string, the payload size is 0x34 bytes, and the snapshot name is @GMT-2006.04.26-04-08-27. The last 2 bytes of the payload are the snapshot strings 16-bit Unicode NULL delimiter.

FRAME 1. Client requests FSCTL_SRV_ENUMERATE_SNAPSHOTS

```
Client -> Server: Command = SMB_COM_NT_TRANSACT
NT IOCTL Function Code 0x00144064 FSCTL_SRV_ENUMERATE_SNAPSHOTS
File ID (Fid) = 16391 (0x4007)
```

FRAME 2. Server response with list of snapshots

```
Server -> Client: Command = SMB_COM_NT_TRANSACT
NT status code = 0x0, STATUS_SUCCESS
Payload contained in Data buffer as defined in section 3.1.5.4:
00090:          01 00 00 00 01 00 00 00 34 00 00 00 40 00   .........4...@.
000A0:  47 00 4D 00 54 00 2D 00 32 00 30 00 30 00 36 00   G.M.T.-.2.0.0.6.
000B0:  2E 00 30 00 34 00 2E 00 32 00 36 00 2D 00 30 00   ..0.4...2.6.-.0.
000C0:  34 00 2E 00 30 00 38 00 2E 00 32 00 37 00 00 00   4...0.8...2.7...
000D0:  00 00
```

The client uses standard SMB commands to access the snapshot. The client also indicates in the header **Flags2** that the name in the request is tokenized with the previous version information. This indicates to the server that the client is accessing a previous version of the path. The server processes the request and returns the path information for the snapshot directory rather than to the current directory.

FRAME 3. Client requests path information for snapshot 2006/04/26 04:08:27 AM

```
  Client -> Server: Command = SMB_COM_TRANSACTION2
  Flags2 Summary = 52231 (0xCC07)
    1100 1100 0000 0111
    .... .1.. .... ....  = File name is tokenized with Previous
  Version Information
  Transact2 function = Query path info
  File name =\@GMT-2006.04.26-04.08.27
  00080:                                         5C 00 40 00
  ……...........\.@.
  00090: 47 00 4D 00 54 00 2D 00 32 00 30 00 30 00 36 00  G.M.T.-.2.0.0.6.
  000A0: 2E 00 30 00 34 00 2E 00 32 00 36 00 2D 00 30 00  ..0.4...2.6.-.0.
  000B0: 34 00 2E 00 30 00 38 00 2E 00 32 00 37 00 00 00  4...0.8...2.7...
```

FRAME 4. Server response with snapshot path information

```
  Server -> Client: Command = SMB_COM_TRANSACTION2
  NT status code = 0x0, STATUS_SUCCESS
  Data bytes = 40 (0x28)
```

Payload contains path information for specified snapshot version

Similar to its behavior during the query path exchange, the client specifies the previous version of the root folder in an open request. The server processes the request and returns an Fid for the specified previous version of the path.

FRAME 5. Client open request for version 2006/04/26 04:08:27 AM on "\"

```
  Client -> Server: Command = SMB_COM_NT_CREATE_ANDX
  Flags2 Summary = 52231 (0xCC07)
    1100 1100 0000 0111
    .... .1.. .... ....  = File name is tokenized with Previous
  Version Information
  Create Disposition = Open:  If exist, Open, else fail
            File name =\@GMT-2006.04.26-04.08.27
```

FRAME 6. Server open root folder and returns Fid

Release: Tuesday, April 6, 2010

```
Server -> Client: Command = SMB_COM_NT_CREATE_ANDX
NT status code = 0x0, STATUS_SUCCESS
File ID (Fid) = 16392 (0x4008)
Create Action = File Opened
```

These similar steps can be used to open a file rather than a directory on a remote volume. In that case, the @GMT token is contained in the relative path, such as \directory\@GMT-2006.04.26-04.08.27\file.txt. This path can be used to query attributes or to open a file. The resulting Fid is used to read its contents.

Likewise, the @GMT token path in the example can be used as part of a TRANS2_FIND_FIRST2 and TRANS2_FIND_NEXT2 to enumerate the contents of the volume at the time of the snapshot.

4.3 Message Signing Example

The following is the sequence of events that is related to SMB message authentication. In the following scenario, as specified in [RFC4178], authentication is used between the client and the server. The client and server are both configured not to require SMB signing; however, both are capable of using SMB signing. This also applies to the figure in section 4.1; however, the parameters significant to signing a negotiation are called out.

1. The client sends an SMB_COM_NEGOTIATE request to the server.

```
Client -> Server: SMB: C negotiate, Dialect = NTLM 0.12
SMB Flags2 contains 0xC843
  1... .... .... .... = Unicode Strings: Strings are Unicode
  .1.. .... .... .... = Error Code Type: Error codes are NT error codes
  ..0. .... .... .... = Execute-Only Reads: Do not permit reads if execute-only
  ...0 .... .... .... = Dfs: Do not resolve pathnames with Dfs
  .... 1... .... .... = Extended security negotiation is supported
  .... .... .1.. .... = Long Names Used
  .... .... .... .0.. = Security signatures are not supported
  .... .... .... ..1. = Extended Attributes: Extended attributes are supported
  .... .... .... ...1 = Long Names Allowed
Security Signature is not set (the value is 00 00 00 00 00 00 00
00).
SECURITY_SIGNATURE: Bit2  (not set)
```

No **SecuritySignature** is generated at this stage.

2. The client receives an SMB_COM_NEGOTIATE response SMB from the server.

```
Server -> Client: SMB: R negotiate, Dialect # = 5
SMB Flags2 contains 0xC853
Binary: 00000000 00000000 11001000 01010011
                              ^      ^ ^

SECURITY_SIGNATURE: Bit2: (not set)
Security Signature is not set (the value is  00 00 00 00 00 00 00 00).
```

No **SecuritySignature** is generated at this stage.

3. The client builds an <u>SMB_COM_SESSION_SETUP_ANDX request</u> SMB and sends it to the server.

   In the SessionSetupAndX SMB, an authentication request, such as an NTLM or NTLMv2 Challenge/Response or a Kerberos ticket, is sent from the client to the server.

   At this stage, the SessionKey is not yet available.

   ```
   Client -> Server: SMB: C session setup & X
   SMB Flags2 contains 0xC807
   Binary: 00000000 00000000 11001000 00000111
                                   ^        ^  ^

   SECURITY_SIGNATURE: Bit2 (set)
   ```

   After the packet is sent by the client, the sequence number is incremented to 1, which is the expected sequence number for the response packet from the server.

4. The server processes the request and sends an <u>SMB_COM_SESSION_SETUP_ANDX response</u> to the client.

   It is possible that multiple roundtrips of SessionSetupAndX can be required to complete a given authentication. If STATUS_MORE_PROCESSING_REQUIRED is returned, then the implementer would return to the previous step and repeat. The following example demonstrates what happens when STATUS_SUCCESS is returned. Similarly, if this authentication was for Anonymous or Guest, then signing would not be activated at this time.

   ```
   Server -> Client: SMB: R session setup & X
   SMB Flags2 contains 0xC807
   Binary: 00000000 00000000 11001000 00000111
                                      ^  ^

   SECURITY_SIGNATURE: Bit2 (set)
   ```

   The server sets the sequence number to 1 for the response packet and generates the **SecuritySignature** as follows.

   The server places the sequence number (1) in the **SecuritySignature** field of the SMB header, and an MD5 hash is performed on the SessionKey + SMB packet. This results in a 16-byte value. The first 8 bytes of the computed hash (AB 44 C4 76 45 84 1A 6A) are placed in the **SecuritySignature** field and sent to the client.

   ```
   00000:  00 11 43 02 26 E6 00 C0 4F 60 2E 45 08 00 45 00   ..C.&f.@O'.E..E.
   00010:  01 78 85 60 40 00 80 32 F6 4B AC 1B 92 B9 AC 1B   .x&'@.,2vK,.9,.
   00020:  92 B7 88 F2 96 BD 00 00 00 14 01 BD 05 48 8B A1   "Fr=.....=.H9!
   00030:  8F 6C C1 3F C0 39 50 18 FF F0 84 70 00 00 00 00   lA?@9P.pp....
   00040:  01 2F FF 53 4D 42 73 00 00 00 00 98 07 C8 00 00   ./SMBs....\.H..
   ```

Release: Tuesday, April 6, 2010

```
00050: >AB 44 C4 76 45 84 1A 6A<00 00 00 00 FF FE 00 08   +DDvE.j....~..
00060:  40 00 04 FF 00 2F 01 00 00 A2 00 04 01 A1 81 9F   @.../..."...!x
00070:  30 81 9C A0 03 0A 01 00 A1 0B 06 09 2A 86 48 82   0S
....!...* H
```

After the server sends the packet, the sequence number is incremented to 2, which is the expected sequence number for the next SMB packet from the client.

5. The client processes the response and obtains the SessionKey.

```
SMB Flags2 contains 0xC807
Binary: 00000000 00000000 11001000 00000111
                              ^   ^


SECURITY_SIGNATURE: Bit2 (set)
```

The expected sequence number is 1 for the response packet from the server.

The client saves the **SecuritySignature** in the response packet. The expected sequence number (1) is placed in the **SecuritySignature** field of the SMB header, and an MD5 hash is performed on the SessionKey SMB packet. This results in a 16-byte value. The first 8 bytes of the computed hash are compared with the one sent by the server (AB 44 C4 76 45 84 1A 6A) to validate the SMB packet. For the SessionKey that is used for signing when Kerberos is used, see [MS-KILE] section 3.1.1.2, Cryptographic Material.

6. The client proceeds further and sends an SMB_COM_TREE_CONNECT_ANDX request SMB.

```
Client -> Server: SMB: C tree connect & X, Share
```

The client sequence number is now incremented. The new value is 2.

The sequence number (2) is placed in the **SecuritySignature** field of the SMB header, and an MD5 hash is performed on the 16-byte SessionKey + SMB packet. This results in a 16-byte value. The first 8 bytes (in this case, A5 B0 43 DC 07 51 0F 8B) are placed in the **SecuritySignature** field in the SMB header and then sent to the server.

```
00000:  00 C0 4F 60 2E 45 00 11 43 02 26 E6 08 00 45 00   .@O'.E..C.&f..E.
00010:  00 98 21 48 40 00 80 32 5B 44 AC 1B 92 B7 AC 1B   .\!H@.,2[D,.",.
00020:  92 B9 C4 70 3D 34 00 00 00 1C 05 48 01 BD C1 3F   9Dp=4.....H.=A?
00030:  C0 39 8B A1 90 9F 50 18 42 EF D0 D6 00 00 00 00   @99!xP.BoPV....
00040:  00 54 FF 53 4D 42 75 00 00 00 00 18 07 C8 00 00   .TSMBu......H..
00050: >A5 B0 43 DC 07 51 0F 8B<00 00 00 00 FF FE 00 08   %0C\.Q.9....~..
00060:  80 00 04 FF 00 54 00 0C 00 01 00 29 00 00 5C 00   ,...T.....)..\.
00070:  5C 00 4D 00 4F 00 48 00 41 00 4B 00 34 00 31 00   \.M.O.H.A.K.4.1.
```

The sequence continues until the session is terminated.

In the case where extended security is not used, the same process is followed. However, the MD5 hash is performed on the 16-byte session key + NTLM challenge response + SMB packet with the appropriate sequence number. The NTLM challenge response is the authentication that is received in the SMB_COM_SESSION_SETUP_ANDX request in the **UnicodePassword** field if NTLM was used for authentication, or in the **OEMPassword** field if LM authentication was used.

4.4 Copy File (Remote to Local)

The following example illustrates the sequence of operations during the copying of a file from a remote location to the local machine. The example assumes that the connection establishment and session management have already taken place.
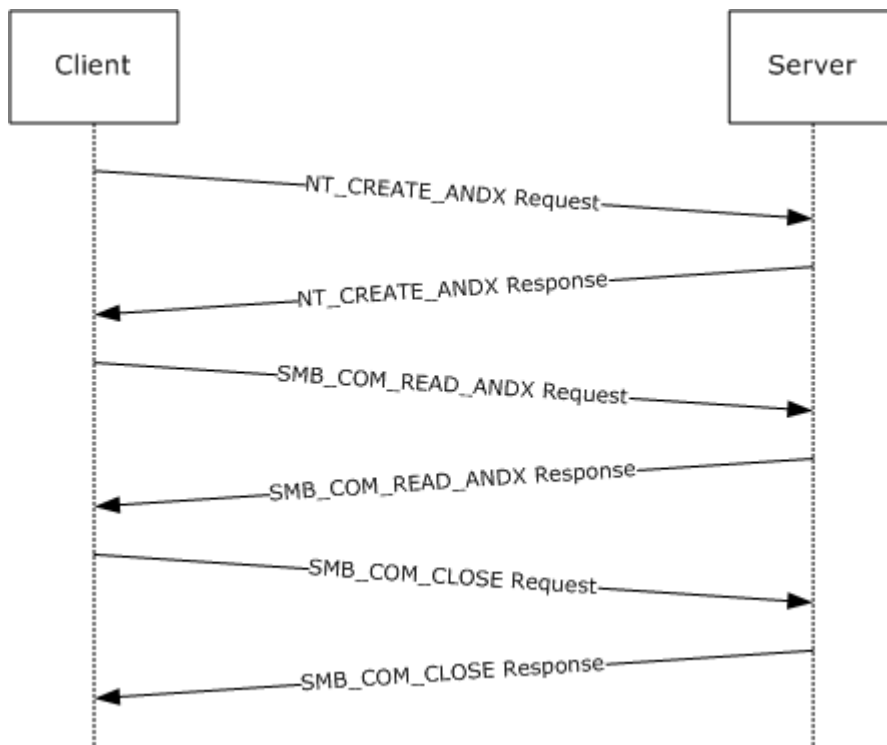

Figure 6: Copy file (remote to local) sequence

In the preceding diagram, the first frame is to open the remote file for read access. The subsequent frames read the data from the file, and then close the file. In between the read and the close, the data is written to the local file.

NT_CREATE_ANDX

```
Client -> Server: SMB: C NT Create Andx, Dialect = NTLM 0.12
SMB: Tree ID      (Tid) = 2049 (0x801)
SMB: Process ID   (Pid) = 3592 (0xE08)
SMB: User ID      (Uid) = 2048 (0x800)
SMB: Multiplex ID (Mid) = 1712 (0x6B0)
SMB: Command = C NT create & X
        SMB: Desired Access = 0x00000089
            SMB: ..............................1 = Read Data Allowed
            SMB: ..............................0. = Write Data Denied
```

Release: Tuesday, April 6, 2010

```
        SMB: ...........................0.. = Append Data Denied
        SMB: ...........................1... = Read EA Allowed
        SMB: ..........................0.... = Write EA Denied
        SMB: .........................0..... = File Execute Denied
        SMB: ........................0...... = File Delete Denied
        SMB: .......................1....... = File Read Attributes Allowed
        SMB: ......................0........ = File Write Attributes Denied
    SMB: NT File Attributes = 0x00000080
        SMB: ..............................0 = Not Read Only
        SMB: .............................0. = Not Hidden
        SMB: ............................0.. = Not System
        SMB: ...........................0... = Not Directory
        SMB: ..........................0.... = Not Archive
        SMB: .........................0..... = Not Device
        SMB: ........................1...... = Normal
        SMB: .......................0....... = Not Temporary
        SMB: ......................0........ = Not Sparse File
        SMB: .....................0......... = Not Reparse Point
        SMB: ....................0.......... = Not Compressed
        SMB: ...................0........... = Not Offline
        SMB: ..................0............ =
CONTENT_INDEXED
        SMB: .................0............. = Not Encrypted
    SMB: File Share Access = 0x00000003
        SMB: ..............................1 = Read allowed
        SMB: .............................1. = Write allowed
        SMB: ............................0.. = Delete not
allowed
    SMB: Create Disposition = Open:  If exist, Open, else fail
    SMB: Create Options = 68 (0x44)
        SMB: ..............................0 = non-directory
        SMB: .............................0. = non-write through
        SMB: ............................1.. = Data must be written to the file
sequentially
        SMB: ...........................0... = intermediate buffering allowed
        SMB: ..........................0.... = IO alerts bits not set
        SMB: .........................0..... = IO non-alerts bit not set
        SMB: ........................1...... = Operation is on a non-directory file
        SMB: .......................0....... = tree connect bit not set
        SMB: ......................0........ = complete if oplocked bit is not set
        SMB: .....................0......... = no EA knowledge bit is not set
        SMB: ....................0.......... = 8.3 filenames bit is not set
        SMB: ...................0........... = random access bit is not set
        SMB: ..................0............ = delete on close bit is not set
        SMB: .................0............. = open by filename
        SMB: ................0.............. = open for backup bit not set
    SMB: File name =\filename.txt
```

NT_CREATE_ANDX Response

```
  Server -> Client: SMB: C NT Create Andx, Dialect = NTLM 0.12
  SMB: Tree ID     (Tid) = 2049 (0x801)
  SMB: Process ID   (Pid) = 3592 (0xE08)
  SMB: User ID     (Uid) = 2048 (0x800)
  SMB: Multiplex ID (Mid) = 1712 (0x6B0)
  SMB: Command = R NT create & X
```

```
                SMB: Oplock Level = Batch
                SMB: File ID (Fid) = 16389 (0x4005)

                SMB: NT File Attributes = 0x00000020
                    SMB: ..............................0 = Not Read Only
                    SMB: .............................0. = Not Hidden
                    SMB: ............................0.. = Not System
                    SMB: ...........................0... = Not Directory
                    SMB: ..........................1.... = Archive
                    SMB: .........................0..... = Not Device
                    SMB: ........................0...... = Not Normal
                    SMB: .......................0....... = Not Temporary
                    SMB: ......................0........ = Not Sparse File
                    SMB: .....................0......... = Not Reparse Point
                    SMB: ....................0.......... = Not Compressed
                    SMB: ...................0........... = Not Offline
                    SMB: ..................0............ = CONTENT_INDEXED
                    SMB: .................0............. = Not Encrypted
```

SMB_COM_READ_ANDX Request

```
  Client -> Server: SMB: C Read Andx, Dialect = NTLM 0.12
          SMB: Tree ID      (Tid) = 2049 (0x801)
          SMB: Process ID   (Pid) = 65279 (0xFEFF)
          SMB: User ID      (Uid) = 2048 (0x800)
          SMB: Multiplex ID (Mid) = 1744 (0x6D0)
  SMB: Command = C read & X
          SMB: File ID (Fid) = 16389 (0x4005)
          SMB: Max count = 1596 (0x63C)
          SMB: Min count = 1596 (0x63C)
          SMB: Bytes left = 1596
```

SMB_COM_READ_ANDX Response

```
  Server -> Client: SMB: R Read Andx, Dialect = NTLM 0.12
          SMB: Tree ID      (Tid) = 2049 (0x801)
          SMB: Process ID   (Pid) = 65279 (0xFEFF)
          SMB: User ID      (Uid) = 2048 (0x800)
          SMB: Multiplex ID (Mid) = 1744 (0x6D0)
  SMB: Command = C read & X
          SMB: Data length = 1596 (0x63C)
          SMB: Data offset = 60 (0x3C)
          SMB: Byte count = 1597

  Data = 00 90 27 D0 C4 6F 00 90 27 66 6D BE 08 00 45 00 ……
```

SMB_COM_CLOSE Request

```
  Client -> Server: SMB: C Close, Dialect = NTLM 0.12
```

```
                 SMB: Tree ID      (Tid) = 2049 (0x801)
                 SMB: Process ID   (Pid) = 65279 (0xFEFF)
                 SMB: User ID      (Uid) = 2048 (0x800)
                 SMB: Multiplex ID (Mid) = 1984 (0x7C0)
      SMB: Command = C Close
                 SMB: File ID (Fid) = 16389 (0x4005)
```

SMB_COM_CLOSE Response

```
   Server -> Client: SMB: R Close, Dialect = NTLM 0.12
          SMB: Tree ID      (Tid) = 2049 (0x801)
          SMB: Process ID   (Pid) = 65279 (0xFEFF)
          SMB: User ID      (Uid) = 2048 (0x800)
          SMB: Multiplex ID (Mid) = 1984 (0x7C0)
```

4.5   Copy File (Local to Remote)

The following example illustrates the sequence of operations while copying a local file to a remote share. The frames do not include the connection establishment or session management, for example.
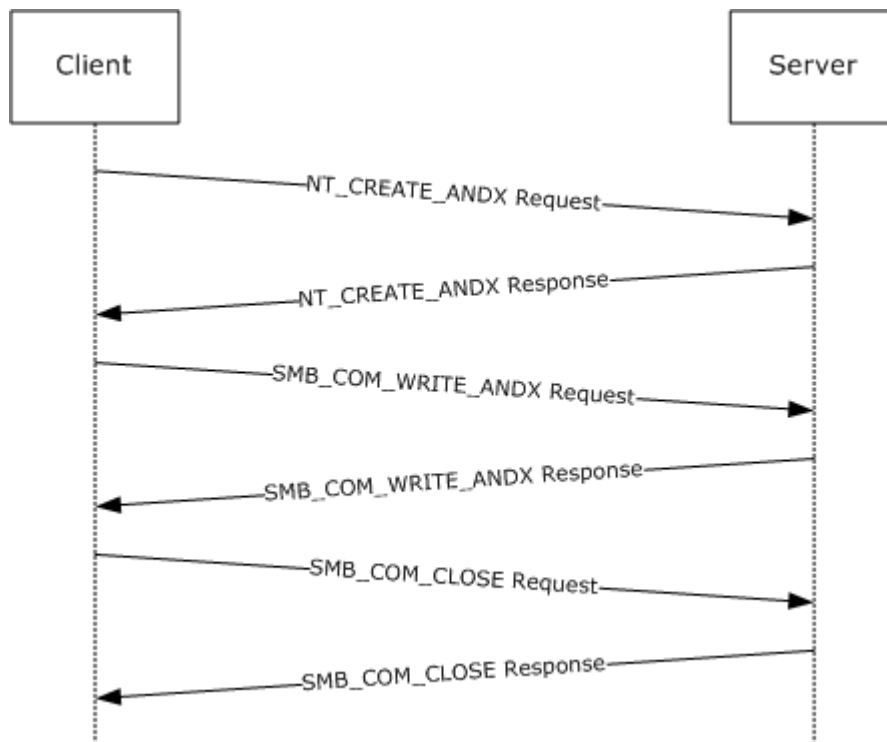
Figure 7: Copy file (local to remote) sequence

In the frames in the preceding figure, the remote file is first created with the
[SMB_COM_NT_CREATE_ANDX request](). The data from the local file is then written to the remote file
and, subsequently, the file is closed.

NT_CREATE_ANDX

```
Client -> Server: SMB: C NT Create Andx, Dialect = NTLM 0.12
      SMB: Tree ID      (Tid) = 2049 (0x801)
      SMB: Process ID   (Pid) = 3592 (0xE08)
      SMB: User ID      (Uid) = 2048 (0x800)
      SMB: Multiplex ID (Mid) = 2288 (0x8F0)
  SMB: Command = C NT create & X
      SMB: Desired Access = 0x00030197
          SMB: ..............................1 = Read Data Allowed
          SMB: .............................1. = Write Data Allowed
          SMB: ............................1.. = Append Data Allowed
          SMB: ...........................0... = Read EA Denied
          SMB: ..........................1.... = Write EA Allowed
          SMB: .........................0..... = File Execute Denied
          SMB: ........................0...... = File Delete Denied
          SMB: .......................1....... = File Read Attributes Allowed
          SMB: ......................1........ = File Write Attributes Allowed
      SMB: NT File Attributes = 0x00000020
          SMB: ..............................0 = Not Read Only
          SMB: .............................0. = Not Hidden
          SMB: ............................0.. = Not System
          SMB: ...........................0... = Not Directory
          SMB: ..........................1.... = Archive
          SMB: .........................0..... = Not Device
          SMB: ........................0...... = Not Normal
          SMB: .......................0....... = Not Temporary
          SMB: ......................0........ = Not Sparse File
          SMB: .....................0......... = Not Reparse Point
          SMB: ....................0.......... = Not Compressed
          SMB: ...................0........... = Not Offline
          SMB: ..................0............ =
  CONTENT_INDEXED
          SMB: .................0............. = Not Encrypted
      SMB: File Share Access = 0x00000000
          SMB: ..............................0 = Read not allowed
          SMB: .............................0. = Write not allowed
          SMB: ............................0.. = Delete not allowed
      SMB: Create Disposition = Overwrite_If:  If exist, open
  and overwrite, else create it
      SMB: Create Options = 68 (0x44)
          SMB: ..............................0 = non-directory
          SMB: .............................0. = non-write through
          SMB: ............................1.. = Data must be written to the file
  sequentially
          SMB: ...........................0... = intermediate buffering allowed
          SMB: ..........................0.... = IO alerts bits not set
          SMB: .........................0..... = IO non-alerts bit not set
          SMB: ........................1...... = Operation is on a non-directory file
          SMB: .......................0....... = tree connect bit not set
          SMB: ......................0........ = complete if oplocked bit is not set
          SMB: .....................0......... = no EA knowledge bit is not set
          SMB: ....................0.......... = 8.3 filenames bit is not set
          SMB: ...................0........... = random access bit is not set
          SMB: ..................0............ = delete on close bit is not set
```

```
             SMB: ..................0............. = open by filename
             SMB: .................0.............. = open for backup bit not set
        SMB: File name =\filename.txt
```

NT_CREATE_ANDX Response

```
  Server -> Client: SMB: R NT Create Andx, Dialect = NTLM 0.12
  SMB: Tree ID      (Tid) = 2049 (0x801)
  SMB: Process ID   (Pid) = 3592 (0xE08)
  SMB: User ID      (Uid) = 2048 (0x800)
  SMB: Multiplex ID (Mid) = 2288 (0x8F0)
  SMB: Command = C NT create & X
        SMB: Oplock Level = Batch
        SMB: File ID (Fid) = 16392 (0x4008)

        SMB: NT File Attributes = 0x00000020
             SMB: ...............................0 = Not Read Only
             SMB: ..............................0. = Not Hidden
             SMB: .............................0.. = Not System
             SMB: ............................0... = Not Directory
             SMB: ...........................1.... = Archive
             SMB: ..........................0..... = Not Device
             SMB: .........................0...... = Not Normal
             SMB: ........................0....... = Not Temporary
             SMB: .......................0........ = Not Sparse File
             SMB: ......................0......... = Not Reparse Point
             SMB: .....................0.......... = Not Compressed
             SMB: ....................0........... = Not Offline
             SMB: ...................0............ = CONTENT_INDEXED
             SMB: ..................0............. = Not Encrypted
```

SMB_COM_WRITE_ANDX Request

```
  Client -> Server: SMB: C Write Andx, Dialect = NTLM 0.12
        SMB: Tree ID      (Tid) = 2049 (0x801)
        SMB: Process ID   (Pid) = 65279 (0xFEFF)
        SMB: User ID      (Uid) = 2048 (0x800)
        SMB: Multiplex ID (Mid) = 2384 (0x950)
  SMB: Command = C read & X
        SMB: File ID (Fid) = 16392 (0x4008)
        SMB: File offset = 0 (0x0)
        SMB: Data length = 1596 (0x63C)
  Data = 00 90 27 66 6D BE 00 90 27 D0 C4 6F 08 00 45 00  ….
```

SMB_COM_WRITE_ANDX Response

```
  Server -> Client: SMB: R Write Andx, Dialect = NTLM 0.12
        SMB: Tree ID      (Tid) = 2049 (0x801)
        SMB: Process ID   (Pid) = 65279 (0xFEFF)
        SMB: User ID      (Uid) = 2048 (0x800)
```

```
            SMB: Multiplex ID (Mid) = 2384 (0x950)
     SMB: Command = C read & X
```

SMB_COM_CLOSE Request

```
  Client -> Server: SMB: C Close, Dialect = NTLM 0.12
         SMB: Tree ID      (Tid) = 2049 (0x801)
         SMB: Process ID   (Pid) = 65279 (0xFEFF)
         SMB: User ID      (Uid) = 2048 (0x800)
         SMB: Multiplex ID (Mid) = 2400 (0x960)
  SMB: Command = C Close
         SMB: File ID (Fid) = 16392 (0x4008)
```

SMB_COM_CLOSE Response

```
  Server -> Client: SMB: R Close, Dialect = NTLM 0.12
         SMB: Tree ID      (Tid) = 2049 (0x801)
         SMB: Process ID   (Pid) = 65279 (0xFEFF)
         SMB: User ID      (Uid) = 2048 (0x800)
         SMB: Multiplex ID (Mid) = 2400 (0x960)
```

4.6 FSCTL SRV COPYCHUNK

The following example refers to the sequence of operations for a file copy in which the source and the destination are on the same server. The FSCTL_SRV_COPYCHUNK (section 2.2.7.2) is used. The following sequence assumes that the SMB connection to the server, SMB session establishment, and other operations have been completed.
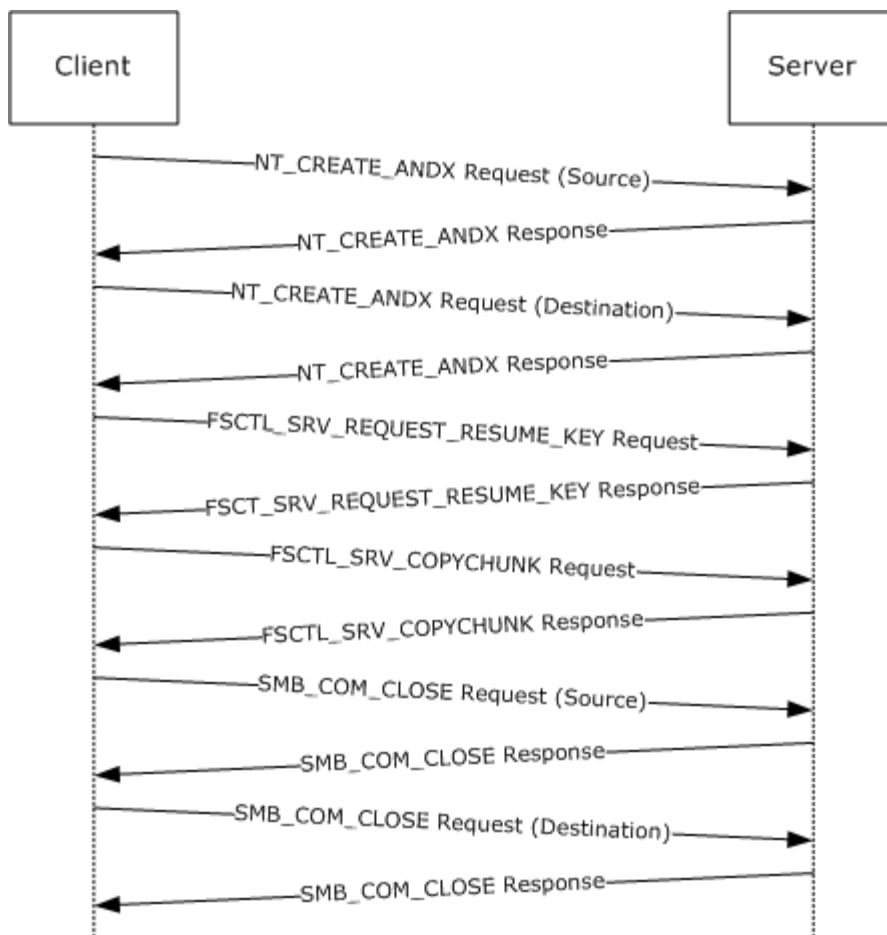
Figure 8: Copy file (from/to same remote server) sequence

The initial step in the preceding sequence is to open the source and the destination file using NT_CREATE_ANDX command. This step is followed by the FSCTL_SRV_REQUEST_RESUME_KEY request. This is sent as an NT_TRANSACT_IOCTL with the file ID of the source file. The server responds with the FSCTL_SRV_REQUEST_RESUME_KEY response (section 2.2.7.2). A 24-byte server copychunk resume key is returned.

NT_CREATE_ANDX Request (Source)

```
Client -> Server: SMB: C NT Create Andx, Dialect = NTLM 0.12
        SMB: Tree ID      (Tid) = 2049 (0x801)
        SMB: Process ID   (Pid) = 3592 (0xE08)
        SMB: User ID      (Uid) = 2048 (0x800)
        SMB: Multiplex ID (Mid) = 384 (0x180)
  SMB: Command = C NT create & X
        SMB: Desired Access = 0x00020089
            SMB: ...........................1 = Read Data Allowed
            SMB: ..........................0. = Write Data Denied
            SMB: .........................0.. = Append Data Denied
            SMB: ........................1... = Read EA Allowed
            SMB: .......................0.... = Write EA Denied
            SMB: ......................0..... = File Execute Denied
```

```
          SMB: ..........................0...... = File Delete Denied
          SMB: .........................1....... = File Read Attributes Allowed
          SMB: ........................0........ = File Write Attributes Denied
     SMB: NT File Attributes = 0x00000000
          SMB: ...............................0 = Not Read Only
          SMB: ..............................0. = Not Hidden
          SMB: .............................0.. = Not System
          SMB: ............................0... = Not Directory
          SMB: ...........................0.... = Not Archive
          SMB: ..........................0..... = Not Device
          SMB: .........................0...... = Not Normal
          SMB: ........................0....... = Not Temporary
          SMB: .......................0........ = Not Sparse File
          SMB: ......................0......... = Not Reparse Point
          SMB: .....................0.......... = Not Compressed
          SMB: ....................0........... = Not Offline
          SMB: ...................0............ =
CONTENT_INDEXED
          SMB: ..................0............. = Not Encrypted
     SMB: File Share Access = 0x00000005
          SMB: ...............................1 = Read allowed
          SMB: ..............................0. = Write not allowed
          SMB: .............................1.. = Delete allowed
     SMB: Create Disposition = Open:  If exist, Open, else fail
     SMB: Create Options = 2097220 (0x200044)
          SMB: ...............................0 = non-directory
          SMB: ..............................0. = non-write through
          SMB: .............................1.. = Data must be written to the
                                                  file sequentially
          SMB: ............................0... = intermediate buffering allowed
          SMB: ...........................0.... = IO alerts bits not set
          SMB: ..........................0..... = IO non-alerts bit not set
          SMB: .........................1...... = Operation is on a non-directory file
          SMB: ........................0....... = tree connect bit not set
          SMB: .......................0........ = complete if oplocked bit is not set
          SMB: ......................0......... = no EA knowledge bit is not set
          SMB: .....................0.......... = 8.3 filenames bit is not set
          SMB: ....................0........... = random access bit is not set
          SMB: ...................0............ = delete on close bit is not set
          SMB: ..................0............. = open by filename
          SMB: .................0.............. = open for backup bit not set
     SMB: File name = sourcefile.txt
```

NT_CREATE_ANDX Response

```
  Server -> Client: SMB: R NT Create Andx, Dialect = NTLM 0.12
       SMB: Tree ID     (Tid) = 2049 (0x801)
       SMB: Process ID  (Pid) = 3592 (0xE08)
       SMB: User ID     (Uid) = 2048 (0x800)
       SMB: Multiplex ID (Mid) = 384 (0x180)
  SMB: Command = R NT create & X
       SMB: Oplock Level = II
       SMB: File ID (Fid) = 16386 (0x4002)

       SMB: NT File Attributes = 0x00000020
          SMB: ...............................0 = Not Read Only
```

```
        SMB: ............................0. = Not Hidden
        SMB: ...........................0.. = Not System
        SMB: ..........................0.... = Not Directory
        SMB: .........................1..... = Archive
        SMB: ........................0...... = Not Device
        SMB: .......................0....... = Not Normal
        SMB: ......................0........ = Not Temporary
        SMB: .....................0......... = Not Sparse File
        SMB: ....................0.......... = Not Reparse Point
        SMB: ..................0............ = Not Compressed
        SMB: .................0............. = Not Offline
        SMB: ................0.............. =
CONTENT_INDEXED
        SMB: ................0.............. = Not Encrypted
    SMB: File type = Disk file or directory
```

NT_CREATE_ANDX Request (Destination)

```
  Client -> Server: SMB: C NT Create Andx, Dialect = NTLM 0.12
        SMB: Tree ID      (Tid) = 2049 (0x801)
        SMB: Process ID   (Pid) = 3592 (0xE08)
        SMB: User ID      (Uid) = 2048 (0x800)
        SMB: Multiplex ID (Mid) = 480 (0x1E0)
  SMB: Command = C NT create & X
        SMB: Desired Access = 0x00030197
        SMB: ..............................1 = Read Data Allowed
        SMB: .............................1. = Write Data Allowed
        SMB: ............................1.. = Append Data Allowed
        SMB: ...........................0... = Read EA Denied
        SMB: ..........................1.... = Write EA Allowed
        SMB: .........................0..... = File Execute Denied
        SMB: ........................0...... = File Delete Denied
        SMB: .......................1....... = File Read Attributes Allowed
        SMB: ......................1........ = File Write Attributes Allowed
        SMB: NT File Attributes = 0x00000020
        SMB: ..............................0 = Not Read Only
        SMB: .............................0. = Not Hidden
        SMB: ............................0.. = Not System
        SMB: ...........................0... = Not Directory
        SMB: ..........................1.... = Archive
        SMB: .........................0..... = Not Device
        SMB: ........................0...... = Not Normal
        SMB: .......................0....... = Not Temporary
        SMB: ......................0........ = Not Sparse File
        SMB: .....................0......... = Not Reparse Point
        SMB: ....................0.......... = Not Compressed
        SMB: ..................0............ = Not Offline
        SMB: .................0............. = CONTENT_INDEXED
        SMB: ................0.............. = Not Encrypted
        SMB: File Share Access = 0x00000000
        SMB: ..............................0 = Read not allowed
        SMB: .............................0. = Write not allowed
        SMB: ............................0.. = Delete not allowed
        SMB: Create Disposition = Overwrite_If:  If exist, open and overwrite,
               else create it
        SMB: Create Options = 68 (0x44)
```

```
        SMB: ..............................0 = non-directory
        SMB: .............................0. = non-write through
        SMB: ............................1.. = Data must be written to the file
sequentially
        SMB: ...........................0... = intermediate buffering allowed
        SMB: ..........................0.... = IO alerts bits not set
        SMB: ..........................0.... = IO non-alerts bit not set
        SMB: ........................1...... = Operation is on a non-directory file
        SMB: .......................0....... = tree connect bit not set
        SMB: ......................0........ = complete if oplocked bit is not set
        SMB: .....................0......... = no EA knowledge bit is not set
        SMB: ....................0.......... = 8.3 filenames bit is not set
        SMB: ...................0........... = random access bit is not set
        SMB: ..................0............ = delete on close bit is not set
        SMB: .................0............. = open by filename
        SMB: ................0.............. = open for backup bit not set
    SMB: File name = destinationfile.txt
```

NT_CREATE_ANDX Response

```
  Server -> Client: SMB: R NT Create Andx, Dialect = NTLM 0.12
        SMB: Tree ID     (Tid) = 2049 (0x801)
        SMB: Process ID  (Pid) = 3592 (0xE08)
        SMB: User ID     (Uid) = 2048 (0x800)
        SMB: Multiplex ID (Mid) = 480 (0x1E0)
  SMB: Command = R NT create & X
        SMB: Oplock Level = Batch
        SMB: File ID (Fid) = 16387 (0x4003)

        SMB: NT File Attributes = 0x00000020
        SMB: ..............................0 = Not Read Only
        SMB: .............................0. = Not Hidden
        SMB: ............................0.. = Not System
        SMB: ...........................0... = Not Directory
        SMB: ..........................1.... = Archive
        SMB: .........................0..... = Not Device
        SMB: ........................0...... = Not Normal
        SMB: .......................0....... = Not Temporary
        SMB: ......................0........ = Not Sparse File
        SMB: .....................0......... = Not Reparse Point
        SMB: ....................0.......... = Not Compressed
        SMB: ...................0........... = Not Offline
        SMB: ..................0............ = CONTENT_INDEXED
        SMB: .................0............. = Not Encrypted
    SMB: File type = Disk file or directory
```

FSCTL_SRV_REQUEST_RESUME_KEY Request

```
  Client -> Server: SMB: C NT Transact, Dialect = NTLM 0.12
  NT IOCTL Function Code 0x00140078 FSCTL_SRV_REQUEST_RESUME_KEY
  File ID (Fid) = 16386 (0x4002)
```

FSCTL_SRV_REQUEST_RESUME_KEY Response

```
Server -> Client: SMB: R NT Transact, Dialect = NTLM 0.12
NT IOCTL Function Code 0x00140078 FSCTL_SRV_REQUEST_RESUME_KEY
File ID (Fid) = 16386 (0x4002)
Key = 2D 0B 00 00 01 00 00 00 59 84 0C 62 1B 84 C6 01 08 0E 00 00 00 00 00 00
ContextLength = 0
```

This is followed by an FSCTL_SRV_COPYCHUNK request. The request uses the resume key generated earlier.

FSCTL_SRV_COPYCHUNK Request

```
Client -> Server: SMB: C NT Transact, Dialect = NTLM 0.12
NT IOCTL Function Code 0x001440F2 FSCTL_SRV_COPYCHUNK
File ID (Fid) = 16387 (0x4003)
Key = 2D 0B 00 00 01 00 00 00 59 84 0C 62 1B 84 C6 01 08 0E 00 00 00 00 00 00
ChunkCount = 1 (01 00 00 00)
Reserved = 0 (00 00 00 00)

List:
    SourceOffset = 0 _(00 00 00 00 00 00 00 00)
    DestinationOffset = 0 (00 00 00 00 00 00 00 00)
    Length = 1731 (3C 06 00 00)
```

FSCTL_SRV_COPYCHUNK Response

```
Server -> Client: SMB: R NT Transact, Dialect = NTLM 0.12
NT IOCTL Function Code 0x001440F2 FSCTL_SRV_COPYCHUNK
File ID (Fid) = 16387 (0x4003)
ChunksWritten = 1 (01 00 00 00)
ChunkBytesWritten = 0 (00 00 00 00)
TotalBytesWritten = 1731 (3C 06 00 00)
```

The final step is to close the source and the destination file with SMB_COM_CLOSE commands.

SMB_COM_CLOSE Request (Source)

```
Client -> Server: SMB: C Close, Dialect = NTLM 0.12
        SMB: Tree ID      (Tid) = 2049 (0x801)
        SMB: Process ID   (Pid) = 65279 (0xFEFF)
        SMB: User ID      (Uid) = 2048 (0x800)
        SMB: Multiplex ID (Mid) = 640 (0x280)
SMB: Command = C Close
        SMB: File ID (Fid) = 16386 (0x4002)
```

SMB_COM_CLOSE Response

```
Server -> Client: SMB: R Close, Dialect = NTLM 0.12
```

```
                SMB: Tree ID      (Tid) = 2049 (0x801)
                SMB: Process ID   (Pid) = 65279 (0xFEFF)
                SMB: User ID      (Uid) = 2048 (0x800)
                SMB: Multiplex ID (Mid) = 640 (0x280)
```

SMB_COM_CLOSE Request (Destination)

```
  Client -> Server: SMB: C Close, Dialect = NTLM 0.12
                SMB: Tree ID      (Tid) = 2049 (0x801)
                SMB: Process ID   (Pid) = 65279 (0xFEFF)
                SMB: User ID      (Uid) = 2048 (0x800)
                SMB: Multiplex ID (Mid) = 656 (0x290)
  SMB: Command = C Close
                SMB: File ID (Fid) = 16387 (0x4003)
```

SMB_COM_CLOSE Response

```
  Server -> Client: SMB: R Close, Dialect = NTLM 0.12
                SMB: Tree ID      (Tid) = 2049 (0x801)
                SMB: Process ID   (Pid) = 65279 (0xFEFF)
                SMB: User ID      (Uid) = 2048 (0x800)
                SMB: Multiplex ID (Mid) = 656 (0x290)
```

4.7   TRANS TRANSACT NMPIPE

The following example illustrates how the TRANS_TRANSACT_NMPIPE is used.
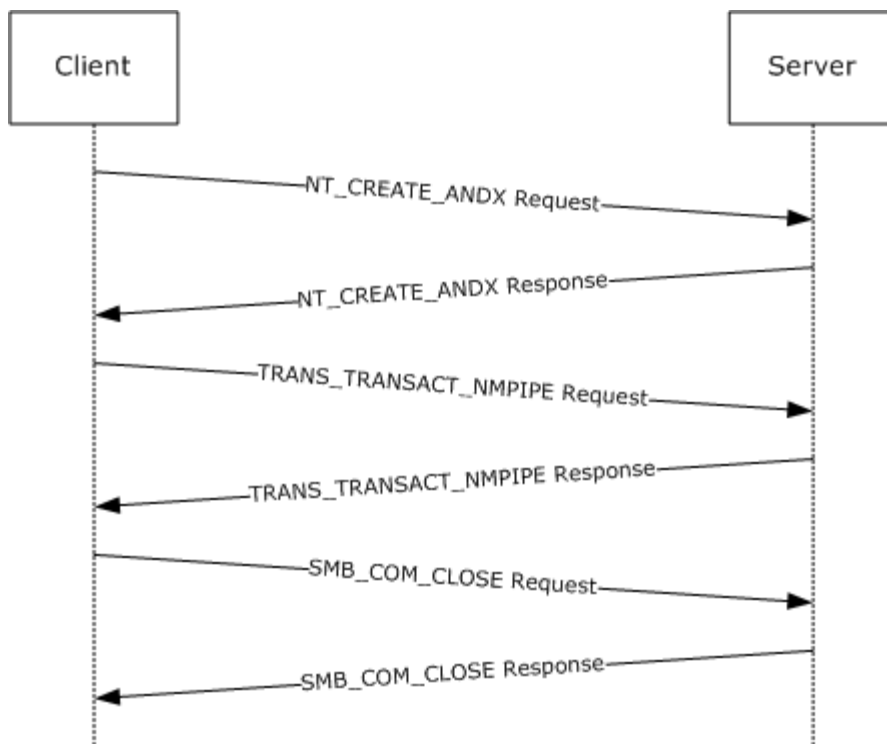
Figure 9: Named pipe request sequence

The first frame contains the NT_CREATE_ANDX request to the named pipe. The TRANS_TRANSACT_NMPIPE is then issued against the file ID assigned in the NT_CREATE_ANDX response.

NT_CREATE_ANDX

```
Client -> Server: SMB: C NT Create Andx, Dialect = NTLM 0.12
        SMB: Tree ID      (Tid) = 2048 (0x800)
        SMB: Process ID   (Pid) = 2292 (0x8F4)
        SMB: User ID      (Uid) = 2048 (0x800)
        SMB: Multiplex ID (Mid) = 4048 (0xFD0)
 SMB: Command = C NT create & X
        SMB: Desired Access = 0x0002019F
            SMB: ..............................1 = Read Data Allowed
            SMB: .............................1. = Write Data Allowed
            SMB: ............................1.. = Append Data Allowed
            SMB: ...........................1... = Read EA Allowed
            SMB: ..........................1.... = Write EA Allowed
            SMB: .........................0..... = File Execute Denied
            SMB: ........................0...... = File Delete Denied
            SMB: .......................1....... = File Read Attributes Allowed
            SMB: ......................1........ = File Write Attributes Allowed
        SMB: NT File Attributes = 0x00000000
            SMB: ..............................0 = Not Read Only
            SMB: .............................0. = Not Hidden
            SMB: ............................0.. = Not System
            SMB: ...........................0... = Not Directory
            SMB: ..........................0.... = Not Archive
```

```
        SMB: ........................0...... = Not Device
        SMB: ........................0....... = Not Normal
        SMB: ........................0........ = Not Temporary
        SMB: .......................0......... = Not Sparse File
        SMB: ......................0.......... = Not Reparse Point
        SMB: .....................0........... = Not Compressed
        SMB: ....................0............ = Not Offline
        SMB: .................0............. =
CONTENT_INDEXED
        SMB: ................0.............. = Not Encrypted
   SMB: File Share Access = 0x00000003
        SMB: ...............................1 = Read allowed
        SMB: ..............................1. = Write allowed
        SMB: .............................0.. = Delete not
allowed

   SMB: Create Disposition = Open:  If exist, Open, else fail
   SMB: Create Options = 4194368 (0x400040)
        SMB: ...............................0 = non-directory
        SMB: ..............................0. = non-write through
        SMB: .............................0.. = non-sequential writing allowed
        SMB: ............................0... = intermediate buffering allowed
        SMB: ...........................0.... = IO alerts bits not set
        SMB: ..........................0..... = IO non-alerts bit not set
        SMB: .........................1...... = Operation is on a non-directory file
        SMB: ........................0....... = tree connect bit not set
        SMB: .......................0........ = complete if oplocked bit is not set
        SMB: ......................0......... = no EA knowledge bit is not set
        SMB: .....................0.......... = 8.3 filenames bit is not set
        SMB: ....................0........... = random access bit is not set
        SMB: ...................0............ = delete on close bit is not set
        SMB: .................0.............. = open by filename
        SMB: ................0............... = open for backup bit not set

   SMB: File name =\srvsvc
```

NT_CREATE_ANDX Response

```
  Server -> Client: SMB: R NT Create Andx, Dialect = NTLM 0.12
       SMB: Tree ID      (Tid) = 2048 (0x800)
       SMB: Process ID   (Pid) = 2292 (0x8F4)
       SMB: User ID      (Uid) = 2048 (0x800)
       SMB: Multiplex ID (Mid) = 4048 (0xFD0)
  SMB: Command = R NT create & X
       SMB: Oplock Level = NONE
       SMB: File ID (Fid) = 16385 (0x4001)

       SMB: NT File Attributes = 0x00000080
            SMB: ..............................0 = Not Read Only
            SMB: .............................0. = Not Hidden
            SMB: ............................0.. = Not System
            SMB: ...........................0... = Not Directory
            SMB: ..........................0.... = Not Archive
            SMB: .........................0..... = Not Device
            SMB: ........................1...... = Normal
            SMB: .......................0....... = Not Temporary
```

```
         SMB: ......................0......... = Not Sparse File
         SMB: .....................0.......... = Not Reparse Point
         SMB: ....................0........... = Not Compressed
         SMB: ...................0............ = Not Offline
         SMB: ..................0............. = CONTENT_INDEXED
         SMB: .................0.............. = Not Encrypted
      SMB: File type = Message mode named pipe
```

SMB_COM_TRANSACTION Request

```
  Client -> Server: SMB: C transact TransactNmPipe, Dialect = NTLM
  0.12
          SMB: Tree ID      (Tid) = 2048 (0x800)
          SMB: Process ID   (Pid) = 2292 (0x8F4)
          SMB: User ID      (Uid) = 2048 (0x800)
          SMB: Multiplex ID (Mid) = 4096 (0x1000)
  SMB: Command = C transact
          SMB: Data bytes = 76 (0x4C)
          SMB: Data offset = 84 (0x54)
          SMB: Setup words
          SMB: Pipe function = Transact named pipe (TransactNmPipe)
          SMB: File ID (Fid) = 16385 (0x4001)
  Data = 00 90 27 66 6D BE 00 90 27 D0 C4 6F 08 00 45 00    ……
```

SMB_COM_TRANSACTION Response

```
  Server -> Client: SMB: R transact TransactNmPipe, Dialect = NTLM
  0.12
          SMB: Tree ID      (Tid) = 2048 (0x800)
          SMB: Process ID   (Pid) = 2292 (0x8F4)
          SMB: User ID      (Uid) = 2048 (0x800)
          SMB: Multiplex ID (Mid) = 4096 (0x1000)
  SMB: Command = R transact
          SMB: Data bytes = 120 (0x78)
          SMB: Data offset = 56 (0x38)

  DATA = 00 90 27 D0 C4 6F 00 90 27 66 6D BE 08 00 45 00 ….
```

SMB_COM_CLOSE Request

```
  Client -> Server: SMB: C Close, Dialect = NTLM 0.12
  SMB: Tree ID      (Tid) = 2048 (0x800)
          SMB: Process ID   (Pid) = 65279 (0xFEFF)
          SMB: User ID      (Uid) = 2048 (0x800)
          SMB: Multiplex ID (Mid) = 4112 (0x1010)
  SMB: Command = C Close
          SMB: File ID (Fid) = 16385 (0x4001)
```

SMB_COM_CLOSE Response

```
Server -> Client: SMB: R Close, Dialect = NTLM 0.12
  SMB: Tree ID      (Tid) = 2048 (0x800)
  SMB: Process ID   (Pid) = 65279 (0xFEFF)
  SMB: User ID      (Uid) = 2048 (0x800)
  SMB: Multiplex ID (Mid) = 4112 (0x1010)
```

5   Security

The following section specifies security considerations for implementers of the Server Message Block (SMB) Protocol.

5.1   Security Considerations for Implementers

The CIFS Protocol contains support for NTLM but lacks support for new authentication protocols. The extensions defined in this document offer support for increased security in remote file and printer access via SMB.

In addition to the NTLM challenge/response authentication support, as specified in [MS-CIFS] section 3.1.5.2, these extensions enable support for Kerberos or any other protocol that can be encapsulated inside the extensible authentication package, as specified in [RFC2743] and [RFC4178].

Extended message signing uses the HMAC_MD5 algorithm, as specified in [RFC2104], to alter the user's session key.

The protocol does not sign oplock break requests from the server to the client if message signing is enabled. This can allow an attacker to affect performance but does not allow an attacker to deny access or alter data.

The algorithm used for message signing has been shown to be subject to collision attacks. See [MD5Collision] for more information.

The new "previous versions" feature potentially allows access to versions of a file that have been deleted or modified. This can provide access to information that was not available without these extensions. However, this access is still subject to the same access checks to which it is normally subject.

5.2   Index of Security Parameters

Security parameter	Section
Signing Key Protection	3.2.5.4
Extended Security - GSS mechanism	3.3.5.3
Extended Security - Maximal User and Guest Rights per Share	3.3.5.4
Authentication Expiration Time	3.3.2.1

6   Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products:

- Microsoft Windows® 2000 operating system

- Windows® XP operating system

- Windows Server® 2003 operating system

- Windows Server® 2003 R2 operating system

- Windows Vista® operating system

- Windows Server® 2008 operating system

- Windows® 7 operating system

- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. The new behavior also applies to subsequent service packs of the product unless otherwise specified.

Unless otherwise specified, any statement of optional behavior in this specification 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 product does not follow the prescription.

<1> Section 1.8: UNIX extensions are not supported by Windows-based SMB clients and servers. The CAP_UNIX capability bit and the Information Level range are reserved to allow third party implementers to collaborate on the definition of these extensions. The development of a common set of extensions has been informally supported by the Storage Networking Industry Association (SNIA). See [SNIA] for SNIA specification on vendor-extension fields.

<2> Section 2.1: The Direct TCP transport can be used by Windows-based SMB clients and servers.

<3> Section 2.1: Windows-based clients and servers use TCP port 445 as the destination TCP port on the SMB server, the well-known port number assigned by IANA to Microsoft-DS.

<4> Section 2.1: The Direct TCP transport header is modeled after the NBT Session Message Packet structure defined in section 4.3.6 of [RFC1002]. The NBT Session Message in the RFC uses a 17-bit length (defined as a two-byte length plus one flag bit--the combination can be read as a 17-bit value in network byte order). Windows systems that support the Direct TCP transport do not exceed the 17-bit limit in the Direct TCP transport header. Handling of the NBT Session Message Packet and the Direct TCP transport header is, therefore, identical in Windows.

<5> Section 2.2: When an error occurs, Windows-based SMB servers return an Error Response message unless specifically required to return data, as specified for named pipe read operations and certain I/O control code requests and other exceptions specified in [MS-CIFS]. Windows-based SMB clients expect that an SMB server returns an Error Response, unless otherwise specified. Windows implementations return data along with these error codes:

- STATUS_MORE_PROCESSING_REQUIRED on a session setup request

- STATUS_BUFFER_OVERFLOW for a read request, IOCTL request, and Query Info request

- STATUS_INVALID_PARAMETER or STATUS_INVALID_VIEW_SIZE for CopyChunk IOCTL request return data along with the header

- STATUS_STOPPED_ON_SYMLINK includes the symbolic link data

- STATUS_BUFFER_TOO_SMALL returns a ULONG containing the required size

<6> Section 2.2.1.2.2: This value is not supported in Windows 2000.

<7> Section 2.2.1.2.2: This value is not supported in Windows 2000.

<8> Section 2.2.1.2.2: This value is not supported in Windows 2000.

<9> Section 2.2.1.2.2: This value is not supported in Windows 2000, Windows Server 2003, Windows Server 2003 R2, Windows XP, Windows Vista, or Windows Server 2008. This value is supported in Windows Server 2008 R2 and Windows 7.

<10> Section 2.2.1.2.2: This value is not supported in Windows 2000, Windows Server 2003, Windows Server 2003 R2, Windows XP, Windows Vista, or Windows Server 2008. This value is supported in Windows Server 2008 R2 and Windows 7.

<11> Section 2.2.1.2.2: This value is not supported in Windows 2000, Windows Server 2003, Windows Server 2003 R2, Windows XP, Windows Vista, or Windows Server 2008. This value is supported in Windows Server 2008 R2 and Windows 7.

<12> Section 2.2.1.2.2: This value is not supported in Windows 2000, Windows Server 2003, Windows Server 2003 R2, Windows XP, Windows Vista, or Windows Server 2008. This value is supported in Windows Server 2008 R2 and Windows 7.

<13> Section 2.2.1.3.1: Windows guarantees uniqueness of FileIds across a single volume.

<14> Section 2.2.2.1: If a client request contains an invalid command code, then Windows 2000 Server and Windows XP server fail the requests by sending an error response with an NTSTATUS code of STATUS_SMB_BAD_COMMAND (ERRSRV/ERRbadcommand). Windows XP SP1, Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, Windows Vista, Windows 7, and Windows Server 2008 R2 servers do not respond to such a request, and do not process further requests on that connection.

<15> Section 2.2.2.3.1: The SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO and SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO Information Levels are available in Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2.

<16> Section 2.2.2.3.5: Pass-through Information Levels on Windows-based servers map directly to native Windows NT Information Classes, as specified in [MS-FSCC] sections 2.4 and 2.5. Windows-based servers do not support setting the following NT Information Levels via the pass-through Information Level mechanism.

Information level	Error code
FileLinkInformation	STATUS_NOT_SUPPORTED
FileMoveClusterInformation	STATUS_NOT_SUPPORTED
FileTrackingInformation	STATUS_NOT_SUPPORTED
FileCompletionInformation	STATUS_NOT_SUPPORTED

Information level	Error code
FileMailslotSetInformation	STATUS_NOT_SUPPORTED

All other Information Levels are passed through to the underlying object store or file system. Refer to [MS-FSCC] sections 2.4 and 2.5 for a further list of Information Levels that are not supported by Windows file systems and the error codes that can be returned.

<17> Section 2.2.2.3.6: These extensions, known as UNIX extensions, are not supported by Windows-based SMB clients and servers. The CAP_UNIX capability bit and the Information Level range specified are reserved to allow third party implementers to collaborate on the definition of these extensions. The development of a common set of extensions has been informally supported by the Storage Networking Industry Association (SNIA).

<18> Section 2.2.2.4: For a detailed listing of possible status codes available on Windows implementations, see [MS-ERREF]. For a list of error codes used by the SMB Version 1.0 Protocol and CIFS Protocol, see [MS-CIFS] section 2.2.2.4.

<19> Section 2.2.3.1: Windows-based servers set the bits in the **Flags2** field with the same value(s) that were sent by the client in the request.

<20> Section 2.2.3.1: Windows-based clients and servers ignore this bit on other messages.

<21> Section 2.2.3.1: Windows-based SMB clients on Windows 2000, Windows XP, and Windows Vista support 32-bit process IDs and use this field when sending the following SMB messages: SMB_COM_NT_CREATE_ANDX and SMB_COM_OPEN_PRINT_FILE. Windows-based SMB clients on Windows 2000, Windows XP, and Windows Vista also support and use this field when sending SMB_COM_TRANSACTION, SMB_COM_TRANSACTION2, and SMB_COM_TRANSACT messages when the server supports the CAP_NT_SMBS bit. The CAP_NT_SMBS bit is set in the **Capabilities** field in the SMB_COM_NEGOTIATE response ([MS-CIFS] section 2.2.4.52.2). Windows 7-based SMB clients do not support 32-bit process IDs and set this field to zero when sending SMB messages. Windows-based SMB servers support 32-bit process IDs when receiving SMB messages.

<22> Section 2.2.4.1.2: Windows-based servers no longer return the FileTypeCommDevice resource type that is listed in [MS-CIFS] section 2.2.4.41.2.

<23> Section 2.2.4.1.2: Windows-based servers do not implement this feature and always set this field to zero.

<24> Section 2.2.4.1.2: Windows-based servers support the notion of a guest account and set this field based on the defined guest account rights on the server.

<25> Section 2.2.4.1.2: Windows 2000, Windows XP, and Windows Vista servers set this field to any value.

<26> Section 2.2.4.2.1: Windows-based servers support MaxCountHigh, but ignore it if set to 0xFFFF.

<27> Section 2.2.4.5.2.1: Windows defaults to a **MaxBufferSize** value of 16,644 bytes on server versions of Windows. Windows defaults to a   **MaxBufferSize** value of 4,356 bytes on client versions of Windows.

<28> Section 2.2.4.5.2.1: Windows Server 2008 does not support SMB_COM_READ_RAW or SMB_COM_WRITE_RAW and disconnects the client by closing the underlying transport connection if either command is received from the client.

<29> Section 2.2.4.5.2.1: Windows Server 2008 does not support SMB_COM_READ_MPX or SMB_COM_WRITE_MPX and disconnects the client by closing the underlying transport connection if either command is received from the client.

<30> Section 2.2.4.5.2.1: Windows-based clients assume that CAP_NT_FIND is set if CAP_NT_SMBS is set.

<31> Section 2.2.4.5.2.1: Windows-based clients and servers take advantage of CAP_INFOLEVEL_PASSTHRU, when available, to prevent the need to map from native file and directory information structures to comparable SMB structures.

<32> Section 2.2.4.5.2.1: With CAP_LARGE_READX enabled, Windows-based servers set this limit to 64 kilobytes. Windows-based clients and servers support CAP_LARGE_READX, which permits file transfers larger than the negotiated MaxBufferSize.

<33> Section 2.2.4.5.2.1: Windows-based clients and servers support CAP_LARGE_WRITEX, which permits file transfers larger than the negotiated MaxBufferSize.

<34> Section 2.2.4.5.2.1: Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 clients and servers support CAP_LWIO.

<35> Section 2.2.4.5.2.1: Windows-based clients and servers do not support CAP_UNIX; therefore, this capability is never set.

<36> Section 2.2.4.5.2.1: Windows-based clients and servers do not support CAP_COMPRESSED_DATA and this capability is never set.

<37> Section 2.2.4.5.2.1: Windows servers do not set the CAP_DYNAMIC_REAUTH flag, even if dynamic re-authentication is supported. On Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, all clients and servers support dynamic re-authentication.

<38> Section 2.2.4.5.2.1: Windows-based clients and servers do not support CAP_PERSISTENT_HANDLES.

<39> Section 2.2.4.5.2.1: Windows-based clients use the **ServerGUID** field.

<40> Section 2.2.4.5.2.1: Windows 2000 Server returns null authentication token in the SecurityBlob field.

<41> Section 2.2.4.5.2.2: Windows-based servers default to a **MaxBufferSize** value of 16,644 bytes. Windows-based clients default to a **MaxBufferSize** value of 4,356 bytes.

<42> Section 2.2.4.5.2.2: Windows-based clients expect 8-byte cryptographic challenges. Windows-based servers provide 8-bit cryptographic challenges.

<43> Section 2.2.4.6.1: Windows-based servers only check for and store a small number of client capabilities:

- CAP_UNICODE

- CAP_LARGE_FILES

- CAP_NT_SMBS

- CAP_NT_FIND

- CAP_NT_STATUS

- CAP_EXTENDED_SECURITY

- CAP_LEVEL_II_OPLOCKS

Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 also check for CAP_DYNAMIC_REAUTH.

<44> Section 2.2.4.6.1: Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 clients set the CAP_DYNAMIC_REAUTH capability bit to indicate to the server that the client supports re-authentication when the Kerberos ticket for the session expires. Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 servers always perform re-authentication when the Kerberos ticket for the session expires, irrespective of whether the client has set the CAP_DYNAMIC_REAUTH flag or not.

<45> Section 2.2.4.6.1: SMB clients set this field based upon the version and service pack level of the Windows operating system. A list of possible values for this field includes the following.

Windows OS version	Native OS string
Windows 2000	Windows 5.0
Windows XP SP2	Windows 2002 Service Pack 2
Windows Server 2003 SP2	Windows Server 2003 3790 Service Pack 2

Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 set this field to an empty string.

<46> Section 2.2.4.6.1: Windows-based SMB clients set this field based upon the version of the Windows operating system. A list of possible values for this field includes the following:

Windows OS version	NativeLanMan string
Windows 2000	Windows 2000 LAN Manager
Windows XP SP2	Windows 2002 5.1
Windows Server 2003	Windows Server 2003 5.2

Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 set this field to an empty string.

<47> Section 2.2.4.6.2: Windows-based SMB servers set this field based upon the version and service pack level of the Windows operating system. The following table includes a list of possible values for this field.

Windows OS version	Native OS string
Windows 2000	Windows 5.0
Windows XP SP2	Windows 2002 Service Pack 2
Windows Server 2003 SP2	Windows Server 2003 3790 Service Pack 2

Windows OS version	Native OS string
Windows Vista Ultimate	Windows Vista Ultimate 6000
Windows Vista Home Basic	Windows Vista 6000
Windows Vista Home Premium	Windows Vista Premium 6000
Windows Server 2008 Datacenter	Windows Server Datacenter 6001 Service Pack 1
Windows 7 Enterprise	Windows 7 Enterprise 7600
Windows 7 Ultimate	Windows 7 Ultimate 7600
Windows Server 2008 R2 Enterprise	Windows Server 2008 R2 Enterprise 7600

<48> Section 2.2.4.6.2: Windows-based SMB servers set this field based upon the version of the Windows operating system. The following table lists possible values for this field.

Windows OS version	NativeLANMan string
Windows 2000	Windows 2000 LAN Manager
Windows XP SP2	Windows 2002 5.1
Windows Server 2003	Windows Server 2003 5.2
Windows Vista Ultimate	Windows Vista Ultimate 6.0
Windows Vista Home Basic	Windows Vista 6.0
Windows Vista Home Premium	Windows Vista Premium 6.0
Windows Server 2008 Datacenter	NativeLANMAN: Windows Server 2008 Datacenter 6.0
Windows 7 Enterprise	Windows 7 Enterprise 6.1
Windows 7 Ultimate	Windows 7 Ultimate 6.1
Windows Server 2008 R2 Enterprise	Windows Server 2008 R2 Enterprise 6.1

<49> Section 2.2.4.7.2: SMB clients on Windows XP, Windows Vista, and Windows 7 client versions of Windows cache directory information if this bit is set on a share. Earlier client versions do not cache the directory information. SMB clients on all server versions of Windows do not cache directory information by default if this bit is set on a share. Caching directory information by SMB clients on Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 can be enabled via a Windows registry setting.

Windows servers set this bit if any of the following are true:

▪ The SHI1005_FLAGS_ALLOW_NAMESPACE_CACHING bit is set for the share, as described in [MS-SRVS] section 2.2.4.29.

▪ NoAliasingOnFilesystem registry key is set to 1 (enabled) on Windows Server 2003 with SP1, Windows Server 2003 R2, Windows Server 2008, or Windows Server 2008 R2.

▪ Short file name generation is disabled in the file system on Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, or Windows Server 2008 R2.

<50> Section 2.2.4.7.2: Windows-based clients and servers support the notion of a guest account and set this field to the access allowed for the guest account.

<51> Section 2.2.4.9.1: Windows 7 and Windows Server 2008 R2 also support two new **CreateOptions** flags:

▪ FILE_OPEN_REQUIRING_OPLOCK (0x00010000)Windows 7 and Windows 2008 R2 servers silently ignore this bit if set in the request. All other Windows-based SMB servers fail requests with the FILE_OPEN_REQUIRING_OPLOCK option set, and return STATUS_INVALID_PARAMETER in the **Status** field of the SMB header in the server response.

▪ FILE_DISALLOW_EXCLUSIVE (0x00020000)Windows 7 and Windows 2008 R2 servers fail requests with this option set, and return STATUS_INVALID_PARAMETER in the **Status** field of the SMB header in the server response.

<52> Section 2.2.4.9.1: Windows-based servers fail the request with STATUS_BAD_IMPERSONATION_LEVEL if the impersonation level is not one of SECURITY_ANONYMOUS, SECURITY_IDENTIFICATION, SECURITY_IMPERSONATION, or SECURITY_DELEGATION.

<53> Section 2.2.4.9.2: Windows-based servers set this field to 0x2A, even though the prescribed 100 bytes (50 WORDs) are present in the **SMB_Parameters.Words** portion of the message.

<54> Section 2.2.4.9.2: Windows-based servers no longer return the FileTypeCommDevice resource type that is listed in [MS-CIFS] section 2.2.4.64.2.

<55> Section 2.2.4.9.2: Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, and Windows Server 2008 set the **VolumeGUID** field to zero; otherwise, this field is uninitialized. The **VolumeGUID** field is ignored by Windows-based SMB clients.

<56> Section 2.2.4.9.2: Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, and Windows Server 2008 set the **FileId** field to zero. The **FileId** field is ignored by Windows-based SMB clients.

<57> Section 2.2.4.9.2: Windows-based servers and clients support the notion of a guest account and set this field appropriately for the defined guest account rights.

<58> Section 2.2.4.9.2: Windows Server 2003, Windows Server 2003 R2, and Windows Server 2008 set this field to zero; otherwise, this field can be sent uninitialized.

<59> Section 2.2.5.1: Windows-based clients never send this request. Windows-based servers fail this request with STATUS_INVALID_PARAMETER.

<60> Section 2.2.5.2: Windows-based clients never send this request.

<61> Section 2.2.6.1.1: Windows-based clients do not issue TRANS2_FIND_FIRST2 requests with the special @GMT-* pattern in the **FileName** field natively. Applications that run on Windows-based clients, however, are allowed to explicitly include the @GMT-* pattern in the pathname that they supply.

<62> Section 2.2.6.1.1: Windows-based clients allow the @GMT-* wildcard to be sent using Information Levels other than SMB_COM_FIND_FILE_BOTH_DIRECTORY_INFO.

<63> Section 2.2.6.4: Support for this subcommand was introduced in Windows 2000.

<64> Section 2.2.6.4.2: Windows-based servers allow only pass-through Information Levels. Windows-based servers respond with STATUS_NOT_SUPPORTED for all other levels.

<65> Section 2.2.7.1: Windows-based servers fail the request with STATUS_BAD_IMPERSONATION_LEVEL if the impersonation level is not one of SECURITY_ANONYMOUS, SECURITY_IDENTIFICATION, SECURITY_IMPERSONATION, or SECURITY_DELEGATION.

<66> Section 2.2.7.2.1: Currently, only Windows Server 2003 with SP1, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 support these new FSCTLs. All Windows-based servers fail requests that contain these FSCTL codes with STATUS_NOT_SUPPORTED.

<67> Section 2.2.7.2.1: A definitive list of Windows FSCTL and IOCTL control codes and their structures (if any) is specified in [MS-FSCC] section 2.3.

<68> Section 2.2.7.2.1: Windows-based clients pass through undefined FSCTL/IOCTL operations to the server. Windows-based servers pass through most FSCTL/IOCTL requests that the server does not understand. In turn, the file system can reject or process these requests. The IOCTL and FSCTL values expected to support specific client features are as specified in [MS-FSCC] section 2.3.

The server blocks certain FSCTL requests by not passing them through to the underlying file system for processing. The following FSCTLs are explicitly blocked by the server and are failed with STATUS_NOT_SUPPORTED.

Name	Value
FSCTL_REQUEST_OPLOCK_LEVEL_1	0x00090000
FSCTL_REQUEST_OPLOCK_LEVEL_2	0x00090004
FSCTL_REQUEST_BATCH_OPLOCK	0x00090008
FSCTL_OPLOCK_BREAK_ACKNOWLEDGE	0x0009000C
FSCTL_OPBATCH_ACK_CLOSE_PENDING	0x00090010
FSCTL_OPLOCK_BREAK_NOTIFY	0x00090014
FSCTL_MOVE_FILE	0x00090074
FSCTL_MARK_HANDLE	0x000900FC
FSCTL_QUERY_RETRIEVAL_POINTERS	0x0009003B
FSCTL_PIPE_ASSIGN_EVENT	0x00110000
FSCTL_GET_VOLUME_BITMAP	0x0009006F
FSCTL_GET_NTFS_FILE_RECORD	0x00090068
FSCTL_INVALIDATE_VOLUMES	0x00090054

Windows does not support USN journal calls because they require a volume handle. The following USN journal calls are also failed with STATUS_NOT_SUPPORTED.

Name	Value
FSCTL_READ_USN_JOURNAL	0x000900BB
FSCTL_CREATE_USN_JOURNAL	0x000900E7

Name	Value
FSCTL_QUERY_USN_JOURNAL	0x000900F4
FSCTL_DELETE_USN_JOURNAL	0x000900F8
FSCTL_ENUM_USN_DATA	0x000900B3

The following FSCTLs are explicitly blocked by Windows Server 2008 R2 and are not passed through to the object store. They are failed with STATUS_NOT_SUPPORTED.

Name	Value
FSCTL_REQUEST_OPLOCK_LEVEL_1	0x00090000
FSCTL_REQUEST_OPLOCK_LEVEL_2	0x00090004
FSCTL_REQUEST_BATCH_OPLOCK	0x00090008
FSCTL_REQUEST_FILTER_OPLOCK	0x0009005C
FSCTL_OPLOCK_BREAK_ACKNOWLEDGE	0x0009000C
FSCTL_OPBATCH_ACK_CLOSE_PENDING	0x00090010
FSCTL_OPLOCK_BREAK_NOTIFY	0x00090014
FSCTL_MOVE_FILE	0x00090074
FSCTL_MARK_HANDLE	0x000900FC
FSCTL_QUERY_RETRIEVAL_POINTERS	0x0009003B
FSCTL_PIPE_ASSIGN_EVENT	0x00110000
FSCTL_GET_VOLUME_BITMAP	0x0009006F
FSCTL_GET_NTFS_FILE_RECORD	0x00090068
FSCTL_INVALIDATE_VOLUMES	0x00090054
FSCTL_READ_USN_JOURNAL	0x000900BB
FSCTL_CREATE_USN_JOURNAL	0x000900E7
FSCTL_QUERY_USN_JOURNAL	0x000900F4
FSCTL_DELETE_USN_JOURNAL	0x000900F8
FSCTL_ENUM_USN_DATA	0x000900B3
FSCTL_QUERY_DEPENDENT_VOLUME	0x000901F0
FSCTL_SD_GLOBAL_CHANGE	0x000901F4
FSCTL_GET_BOOT_AREA_INFO	0x00090230
FSCTL_GET_RETRIEVAL_POINTER_BASE	0x00090234

Name	Value
FSCTL_SET_PERSISTENT_VOLUME_STATE	0x00090238
FSCTL_QUERY_PERSISTENT_VOLUME_STATE	0x0009023C
FSCTL_REQUEST_OPLOCK	0x00090240
FSCTL_TXFS_MODIFY_RM	0x00098144
FSCTL_TXFS_QUERY_RM_INFORMATION	0x00094148
FSCTL_TXFS_ROLLFORW ARD_REDO	0x00098150
FSCTL_TXFS_ROLLFORWARD_UNDO	0x00098154
FSCTL_TXFS_START_RM	0x00098158
FSCTL_TXFS_SHUTDOWN_RM	0x0009815C
FSCTL_TXFS_READ_BACKUP_INFORMATION	0x00094160
FSCTL_TXFS_WRITE_BACKUP_INFORMATION	0x00098164
FSCTL_TXFS_CREATE_SECONDARY_RM	0x00098168
FSCTL_TXFS_GET_METADATA_INFO	0x0009416C
FSCTL_TXFS_GET_TRANSACTED_VERSION	0x00094170
FSCTL_TXFS_SAVEPOINT_INFORMATION	0x00098178
FSCTL_TXFS_CREATE_MINIVERSION	0x0009817C
FSCTL_TXFS_TRANSACTION_ACTIVE	0x0009418C
FSCTL_TXFS_LIST_TRANSACTIONS	0x000941E4
FSCTL_TXFS_READ_BACKUP_INFORMATION2	0x000901F8
FSCTL_TXFS_WRITE_BACKUP_INFORMATION2	0x00090200

<69> Section 2.2.7.2.1: Currently, only Windows Server 2003 with SP1, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 support this FSCTL. All other Windows-based servers fail the request with STATUS_NOT_SUPPORTED.

<70> Section 2.2.7.2.1: Currently, only Windows Server 2003 with SP1, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 servers support this FSCTL. All other Windows-based servers fail the request with STATUS_NOT_SUPPORTED.

<71> Section 2.2.7.2.1: Windows-based Servers return a STATUS_INVALID_PARAMETER error if the value of MaxDataCount is less than 0x10 bytes.

<72> Section 2.2.7.2.1.1: Windows-based clients do not initialize the **Reserved** field to zero.

<73> Section 2.2.7.2.2.2: Windows based servers include the **Context** field but do not initialize it. Given that they also always set the **ContextLength** field to zero, it is not possible to determine what values a server uses for this field. However, this field is treated as junk bytes at the end of the message and is safely ignored by clients.

The SMB_FIND_FILE_ID_FULL_DIRECTORY_INFO Information Level was introduced in Windows Server 2003 and is not present in Windows server 2000 and Windows XP.

The SMB_FIND_FILE_ID_BOTH_DIRECTORY_INFO Information Level was introduced in Windows Server 2003 and is not present in Windows server 2000 and Windows XP.

The following attribute flags are removed by the Windows Server 2008 R2 SMB server before sending the attribute data block to the client:

- FILE_SUPPORTS_TRANSACTIONS

- FILE_SUPPORTS_OPEN_BY_FILE_ID

Windows-based servers return a slightly altered form of the FILE_LINK_INFORMATION structure (specified in [MS-FSCC] section 2.4.21), called an SMB_FILE_LINK_INFORMATION structure, for the FileLinkInformation Information Class.

FileLinkInformation

The FileLinkInformation Information Class is used to create an NT file system (**NTFS**) hard link to an existing file.

The SMB_FILE_LINK_INFORMATION data element is as follows.

```
SMB_FILE_LINK_INFORMATION
  {
  BOOLEAN ReplaceIfExists;
  UCHAR   Reserved[3];
  ULONG   RootDirectory;
  ULONG   FileNameLength;
  WCHAR   FileName[FileNameLength/2];
  }
```

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
ReplaceIfExists								Reserved																							
RootDirectory																															
FileNameLength																															
FileName (variable)																															
...																															

ReplaceIfExists (1 byte): A BOOLEAN value. If a target exists and ReplaceIfExists is FALSE, then the operation must be failed with STATUS_OBJECT_NAME_COLLISION. If the target exists and ReplaceIfExists is TRUE, then the server should attempt to replace the target.

Reserved (3 bytes): This field should be set to zero by the client and must be ignored by the server.

RootDirectory (4 bytes): A 32-bit unsigned integer that contains the file handle for the directory where the link is to be created. This value must be zero.

FileNameLength (4 bytes): A 32-bit unsigned integer that contains the length, in bytes, of the FileName field.

FileName (variable): A sequence of UNICODE characters that contain the name to be assigned to the newly-created link. When working with the **FileName** field, use **FileNameLength** to determine the length of the file name rather than assuming the presence of a trailing null delimiter. If the **RootDirectory** field is NULL and the link is to be created in a different directory from the file that is being linked to, then this field specifies the full path name for the link to be created; otherwise, it specifies only the file name.

<78> Section 3.1.4.1: Windows-based SMB servers use the entire session key for signing if the session key length is equal to or greater than 16, and pad the session key with zero up to 16 bytes if the session key length is less than 16.

<79> Section 3.1.5.1: Windows-based SMB servers use the entire session key for signing if the session key length is equal to or greater than 16, and pad the session key with zero up to 16 bytes if the session key length is less than 16.

<80> Section 3.2.3: **Client.SupportsExtendedSecurity** is TRUE for Windows- based clients.

<81> Section 3.2.4.1.1: Windows XP, Windows Vista, Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, Windows 7, and Windows Server 2008 R2 clients scan pathnames for previous version tokens and set the SMB_FLAGS2_REPARSE_PATH flag if a token is found.

<82> Section 3.2.4.2.2: Windows XP SP3-based clients and Windows Vista-based SMB clients set the SMB_FLAGS2_SMB_SECURITY_SIGNATURE_REQUIRED bit in the **Flags2** field of the SMB header of a Session Setup request.

<83> Section 3.2.4.2.4: Windows -based SMB clients use the same connection for all authentications other than terminal services. Connections configured for terminal services use one connection per user.

<84> Section 3.2.4.2.4: Windows no longer supports the "Disabled" state. Windows clients and servers only support "Disabled Unless Required". This change has been implemented in Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

<85> Section 3.2.4.2.4: Windows-based clients implement this option

<86> Section 3.2.4.2.4:

- Windows-based clients support extended security.

- Windows systems implement the first option that is previously described.

<87> Section 3.2.4.2.4.1: Windows-based SMB clients are configured by default to not send plain text passwords. Sending plain text passwords can be configured via a registry setting.

<88> Section 3.2.4.2.5: Windows-based servers establish a new tree connect for each session even if the target share is the same.

<89> Section 3.2.4.2.5: Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 request **Client.Session.SessionKey** protection.

<90> Section 3.2.4.2.5: Windows-based clients request extended information responses.

<91> Section 3.2.4.3: Windows -based clients issue an SMB_COM_NT_CREATE_ANDX request for the NT LM 0.12 dialect for which all of the extensions here are described.

<92> Section 3.2.4.3.2: Windows-based clients do not use this flag.

<93> Section 3.2.4.4: Windows-based clients set the **Timeout** field to 0xFFFFFFFF on pipe reads.

<94> Section 3.2.4.4.1: Windows-based clients issue large reads if the server supports them.

<95> Section 3.2.4.6: Windows-based clients send these requests to the server regardless of the Information Level provided in the request.

<96> Section 3.2.4.7: Windows-based clients use pass-through Information Levels if available.

<97> Section 3.2.4.7: Windows-based servers return a STATUS_INVALID_PARAMETER error if the **FID** field in the client request is on an SMB share that does not support streams.

<98> Section 3.2.4.7: Windows-based servers return a STATUS_INVALID_PARAMETER error if the **FileName** field in the client request is on an SMB share that does not support streams.

<99> Section 3.2.4.8: Windows-based clients use pass-through Information Levels if available.

<100> Section 3.2.4.9: Windows-based clients use pass-through Information Levels, if available.

<101> Section 3.2.4.11.1: Windows XP, Windows Vista, and Windows 7 clients use this FSCTL. Windows 2000-based clients can use this FSCTL if the previous versions' down-level application is installed on them.

<102> Section 3.2.4.12: Windows XP SP3, Windows Server 2003 with SP1, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 clients support DFS.

<103> Section 3.2.5.1: Windows-based servers fail incorrectly signed packets with ACCESS_DENIED but do not terminate the connection.

<104> Section 3.2.5.2: Windows-based SMB servers support Extended Security. They all are configured to use SPNEGO, as specified in [RFC4178], as their GSS authentication protocol. Windows operating systems that use extended security send a GSS token (or fragment) if their SPNEGO implementation supports it. See [RFC4178] for details on Windows behavior.

<105> Section 3.2.5.3: Windows Vista SP1, Windows Server 2008, Windows 7, and Windows Server 2008 R2 servers fail a non-extended security session setup request with STATUS_INVALID_PARAMETER if the registry key is either missing or set to zero.

<106> Section 3.2.5.5: Windows-based servers will always set this value to 0x2A (42) when using an extended response.

<107> Section 3.3.2.1: The default expiration time is eight hours for the Windows implementation of Kerberos.

<108> Section 3.3.3: **SupportsExtendedSecurity** is TRUE for Windows -based clients

<109> Section 3.3.4.1.1: Windows servers do not respond with an OS/2 error on the wire even if SMB_FLAGS2_NT_STATUS is set in the client request (see [MS-CIFS] section 2.2.3.1). If the negotiated dialect is DOS LANMAN 2.0, DOS LANMAN 2.1, or prior to LANMAN 1.0, an ERROR_GEN_FAILURE error is returned. Otherwise, the following table lists the corresponding DOS error that is returned:

OS/2 Error	DOS Error
STATUS_OS2_INVALID_LEVEL	ERROR_INVALID_LEVEL
STATUS_OS2_EA_LIST_INCONSISTENT	ERROR_EA_LIST_INCONSISTENT
STATUS_OS2_NEGATIVE_SEEK	ERROR_NEGATIVE_SEEK
STATUS_OS2_NO_MORE_SIDS	ERROR_NO_MORE_SEARCH_HANDLES
STATUS_OS2_EAS_DIDNT_FIT	ERROR_EAS_DIDNT_FIT
STATUS_OS2_EA_ACCESS_DENIED	ERROR_EA_ACCESS_DENIED
STATUS_OS2_CANCEL_VIOLATION	ERROR_CANCEL_VIOLATION
STATUS_OS2_ATOMIC_LOCKS_NOT_SUPPORTED	ERROR_ATOMIC_LOCKS_NOT_SUPPORTED
STATUS_OS2_CANNOT_COPY	ERROR_CANNOT_COPY

<110> Section 3.3.5.1: Windows-based servers respond to incorrectly signed packets with a STATUS_ACCESS_DENIED status but do not terminate the connection.

<111> Section 3.3.5.1.1: SMB servers on Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 support the SMB_FLAGS2_REPARSE_PATH flag and previous version access. SMB server on Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 parses paths when the flag is not set but only when configured to do so. This flag is used to expose the previous version logic to applications that run on clients whose SMB client does not understand the SMB_FLAGS2_REPARSE_PATH flag and does not set it appropriately.

<112> Section 3.3.5.2: Windows-based SMB servers support Extended Security, and are configured to use SPNEGO (as specified in [RFC4178]) as their GSS authentication protocol. Windows operating systems that use extended security send a GSS token (or fragment) if their SPNEGO implementation supports it. For details on Windows behavior, see [RFC4178].

<113> Section 3.3.5.5: SMB servers on Windows 2000 Server, Windows Server 2003, Windows Server 2008, and Windows Server 2008 R2 use extended responses if the client requests them.

<114> Section 3.3.5.5: Windows-based servers set the **FileStatusFlags** using the following mapping of output elements specified in [MS-FSA] section 3.1.5.1:

- NO_EAS is set if the returned **Open.File.ExtendedAttributesLength** is zero, otherwise it is not set.

- NO_SUBSTREAMS is set if the returned **Open.File.StreamList** is less than or equal to one, otherwise it is not set.

- NO_REPARSETAG is set if the returned Open.File.ReparseTag is empty, otherwise it is not set.

<115> Section 3.3.5.5: NTFS supports streams. **FAT** and FAT32 file systems do not support streams.

<116> Section 3.3.5.5: SMB servers on Windows 2000 Server, Windows Server 2003, Windows Server 2008, and Windows Server 2008 R2 return zero for the **VolumeGUID** and **FileId**. All other Windows-based servers set the **VolumeGUID** and **FileId** fields using the following mapping of output elements, specified in [MS-FSA] section 3.1.5.1:

- **VolumeGUID** is set to the returned **Open.File.Volume.VolumeId**.

- **FileId** is set to the returned **Open.File.FileId**.

<117> Section 3.3.5.5: Windows-based servers set the **MaximalAccessRights** and **GuestMaximalAccessRights** fields using the following mapping of output elements, specified in [MS-FSA] section 3.1.5.1:

- **MaximalAccessRights** is set to the returned **Open.GrantedAcess**.

- **GuestMaximalAccessRights** is set to the returned **Open.GuestGrantedAcess**.

<118> Section 3.3.5.5: Windows-based SMB servers allow the following options on a named pipe:

- FILE_WRITE_THROUGH

- FILE_SYNCHRONOUS_IO_ALERT

- FILE_SYNCHRONOUS_IO_NONALERT

    FILE_SYNCHRONOUS_IO_ALERT and FILE_SYNCHRONOUS_IO_NONALERT are not combined.

<119> Section 3.3.5.6: Windows servers use extended responses if the client requests them.

<120> Section 3.3.5.6: Windows-based servers set the **MaximalAccessRights** and **GuestMaximalAccessRights** fields using the following mapping of output elements specified in [MS-FSA]section 3.1.5.1:

- **MaximalAccessRights** is set to the returned **Open.GrantedAcess**.

- **GuestMaximalAccessRights** is set to the returned **Open.GuestGrantedAcess**.

<121> Section 3.3.5.7: Windows-based SMB servers allow for large reads of 64 kilobytes, but no larger. Requests for larger values return, at most, 64 kilobytes.

<122> Section 3.3.5.9.1: Windows behavior for each Information Class is specified in each Information Class' corresponding subsection of either [MS-FSA] sections 3.1.5.11 or 3.1.5.12.

<123> Section 3.3.5.9.2: Windows servers support these new Information Levels for directory queries.

<124> Section 3.3.5.9.2: Windows Server 2003, Windows Server 2003 R2, and Windows Server 2008 support previous versions but do not support this method of enumerating them, by default. This feature can be configured to be active by the administrator. The purpose is to allow an application (on a client that does not support the IOCTL command) to have a method of enumerating the previous versions.

<125> Section 3.3.5.9.6: If the requested Information Class is FileRenameInformation, then the following validation is performed:

- If **RootDirectory** is not NULL, then the request must be failed with STATUS_INVALID_PARAMETER.

- If the file name pointed to by the *FileName* parameter of the FILE_RENAME_INFORMATION structure contains a separator character, then the request should be failed with STATUS_NOT_SUPPORTED.

If the server file system does not support this **Information Level**, then it must fail the request with STATUS_NOT_SUPPORTED. Otherwise, it must attempt to apply the attributes to the target file and return the success or failure code in the response.

<126> Section 3.3.5.9.7: Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, and Windows Server 2008 do not break a batch oplock when processing a TRANS2_SET_PATH_INFORMATION request. Windows Server 2008 R2 breaks a batch oplock when processing the request.

<127> Section 3.3.5.10.1.3: Windows Server 2003 with SP1 default values and limits are as follows:

- Fail the request and set the status to STATUS_IO_TIMEOUT.

- Return the number of written chunks as specified in section 2.2.7.2.2.

- The maximum number of chunks that the server accepts to be copied in a single request is 256.

- The maximum amount of data that the server accepts to be copied in a single chunk is one megabyte.

- The maximum amount of data that the server accepts to be copied in a single request is 16 megabytes.

- The total size (lengths) of all of the chunks combined must be greater than or equal to 16 megabytes.

- The time-out for server-side data copy operations to complete is 25 seconds.

<128> Section 3.3.5.10.2: Windows-based servers request quota information from the object store, as specified in [MS-FSA] section 3.1.5.11.24 if **Server.Open** is a file. If **Server.Open** is on a directory, then the processing follows with the following mapping of input elements:

- **Open** is an Open of **Server.Open.TreeConnect.Share.LocalPath** for the **Server.Open** indicated by the **SMB_Parameters.Words.FID** field of the request.

- **OutputBufferSize** is the **SMB_Parameters.Words.MaxDataCount** field of the request.

- **ReturnSingle** is the **NT_Trans_Parameters.ReturnSingleEntry** field of the request.

- **RestartScan** is the **NT_Trans_Parameters.RestartScan** field of the request.

- **SidList** is the **NT_Trans_Data.SidList** field of the request.

The returned **Status** is copied into the **SMB_Header.Status** field of the response. If the operation is successful, then the following additional mapping of output elements applies:

- **OutputBuffer** is copied into the **NT_Trans_Data** field of the response.

- **ByteCount** is copied into the **SMB_Parameters.TotalDataCount** field of the response.

If quotas are disabled, then NTFS may return STATUS_SUCCESS with the UserQuotaData.QSize set to zero and with both the UserQuotaData.QuotaThreshhold field and the UserQuotaData.QuotaLimit field set to 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.

<129> Section 3.3.5.10.3: Windows-based servers set the quota information on the object store, as specified in [MS-FSA] section 3.1.5.14.10 if **Server.Open** is on a file. If **Server.Open** is on a directory, then processing follows as specified elsewhere in [MS-FSA] section 3.1.5 with the following mapping of input elements:

- **Open** is an Open of **Server.Open.TreeConnect.Share.LocalPath** for the **Server.Open** indicated by the **SMB_Parameters.Words.FID** field of the request.

- **OutputBufferSize** is the **SMB_Parameters.Words.MaxDataCount** field of the request.

- **ReturnSingle** is the **NT_Trans_Parameters.ReturnSingleEntry** field of the request.

- **RestartScan** is the **NT_Trans_Parameters.RestartScan** field of the request.

- **SidList** is the **NT_Trans_Data.SidList** field of the request.

The returned **Status** is copied into the **SMB_Header.Status** field of the response. If the operation is successful, then the following additional mapping of output elements applies:

- **OutputBuffer** is copied into the **NT_Trans_Data** field of the response.

- **ByteCount** is copied into the **SMB_Parameters.TotalDataCount** field of the response.

If the volume is capable of supporting disk quotas but has the feature disabled, then a STATUS_INVALID_DEVICE_REQUEST is returned.

7   Change Tracking

This section identifies changes made to [MS-SMB] protocol documentation between March 2010 and April 2010 releases. Changes are classed as major, minor, or editorial.

Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.

- An extensive rewrite, addition, or deletion of major portions of content.

- A protocol is deprecated.

- The removal of a document from the documentation set.

- Changes made for template compliance.

Minor changes do not affect protocol interoperability or implementation. Examples are updates to fix technical accuracy or ambiguity at the sentence, paragraph, or table level.

Editorial changes apply to grammatical, formatting, and style issues.

No changes means that the document is identical to its last release.

Major and minor changes can be described further using the following revision types:

- New content added.

- Content update.

- Content removed.

- New product behavior note added.

- Product behavior note updated.

- Product behavior note removed.

- New protocol syntax added.

- Protocol syntax updated.

- Protocol syntax removed.

- New content added due to protocol revision.

- Content updated due to protocol revision.

- Content removed due to protocol revision.

- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.

- Protocol syntax removed due to protocol revision.

- New content added for template compliance.

- Content updated for template compliance.

- Content removed for template compliance.

- Obsolete document removed.

Editorial changes always have the revision type "Editorially updated."

Some important terms used in revision type descriptions are defined as follows:

Protocol syntax refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.

Protocol revision refers to changes made to a protocol that affect the bits that are sent over the wire.

Changes are listed in the following table. If you need further information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Revision Type
1.1 Glossary	Specified the definition of security context.	N	Content update.
1.1 Glossary	47948 Removed the term "globally unique identifier (GUID)" from [MS-GLOS].	Y	Content update.
1.2.2 Informative References	Added [MSDFS] reference.	N	New content added.
1.2.2 Informative References	Removed the reference [MS-SECO] and added [MS-WSO].	Y	Content update.
2.2.1.1.1 Pathname Extensions	48289 Added sentences to distinguish between the examples for intermediate and final path components.	N	New content added.
2.2.1.1.1 Pathname Extensions	48289 Revised the description and usage of snippets for previous versions.	Y	Content update.
2.2.1.1.1 Pathname Extensions	48289 Removed duplicate code example.	N	Content removed.
2.2.1.2.1 Extended File Attribute (SMB_EXT_FILE_ATTR) Extensions	48288 Removed "immediate" from "ATTR_OFFLINE" meaning.	N	Content removed.
2.2.1.2.2 File System Attribute Extensions	47295 Modified the link to the list of file system attributes in [MS-CIFS].	N	Content update.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Revision Type
2.2.1.3 Unique Identifiers	47948 Added a reference to GUID defined in [MS-DTYP].	Y	Content update.
2.2.1.3.2 VolumeGUID Generation	47066 Updated the normative language of the constraints for VolumeGUIDs.	Y	Content update.
2.2.2.2 Transaction Subcommand Codes	47306 Removed the product behavior note in the description of NT_TRANSACT_CREATE2.	Y	Product behavior note removed.
2.2.2.2 Transaction Subcommand Codes	48004 Updated the description of NT_TRANSACT_CREATE2.	N	Content update.
2.2.2.4 SMB Error Classes and Codes	47053 Added the error code STATUS_STOPPED_ON_SYMLINK.	N	Content update.
2.2.3.1 SMB Header Extensions	48288 Removed "immediately" from "SMB_FLAGS2_SMB_SECURITY_SIGNATURE_REQUIRED" meaning.	N	Content removed.
2.2.4.1.1 Client Request Extensions	47308 Updated the fields in the description of SMB_OPEN_QUERY_INFORMATION.	Y	Content update.
2.2.4.1.2 Server Response Extensions	47909 Updated the description of the ServerFID field.	Y	Content update.
2.2.4.1.2 Server Response Extensions	47917 Updated the product behavior note in the description of ByteCount.	Y	Product behavior note updated.
2.2.4.2.1 Client Request Extensions	47918 Moved the content of the product behavior note in Timeout to the description of the Timeout field.	Y	Product behavior note removed.
2.2.4.5.2.1 Extended Security Response	48066 Changed the SMB_Parameter.SMB_Data.ServerGUID data type from USHORT to GUID.	N	Content update.
2.2.4.5.2.1 Extended Security Response	48589 Updated the description of CAP_RPC_REMOTE_APIS, CAP_DFS, and CAP_INFOLEVEL_PASSTHRU.	N	Content update.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Revision Type
2.2.4.9.1 Client Request Extensions	Changed the reference [MS-SECO] to [MS-WSO].	N	Content update.
2.2.7.1 NT_TRANSACT_CREATE (0x0001) Extensions	Changed the reference [MS-SECO] to [MS-WSO].	N	Content update.
2.2.7.2.1 Client Request Extensions	47063 Clarified the value received in the CopychunkResumeKey field.	Y	Content update.
3.1.5.1 Receiving Any Message	48596 Updated the normative language for generating a hash.	Y	Content update.
3.2.1.2 Per SMB Connection	48066 Removed "universal" from "Client.Connection.Server.GUID" description.	N	Content removed.
3.2.4.2.4 User Authentication	Updated the reference for Generic Security Services (GSS) authentication.	N	Content update.
3.2.4.2.4 User Authentication	48595 Updated the normative language for Client.MessageSigningPolicy.	Y	Content update.
3.2.4.7 Application Requests Querying File Attributes	48589 Updated the description of file information classes.	N	Content update.
3.2.4.8 Application Requests Setting File Attributes	48589 Updated the description of file information classes.	N	Content update.
3.2.4.9 Application Requests Querying File System Attributes	48589 Updated the description of file information classes.	N	Content update.
3.2.4.11 Application Requests Sending an I/O Control to a File or Device	48589 Updated the description of commonly used FSCTLs.	N	Content update.
3.2.5.4 Receiving an SMB_COM_TREE_CONNECT_AN DX Response	Updated the user new session key to be 16 bytes.	N	Content update.
3.3.1.1 Global	Added the Server.IsDfsCapable section.	Y	New content added.
3.3.1.2 Per Share	43711 Added a table that specifies how the elements of the	N	Content update.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Revision Type
	Server.Share.SecurityDescriptor structure map into the shared Share object.		
3.3.1.2 Per Share	Added Server.Share.IsDfs paragraph.	N	New content added.
3.3.3 Initialization	Added Server.IsDfsCapable and Server.Share.IsDfs paragraphs.	Y	New content added.
3.3.3 Initialization	Tagged two values as list items.	N	Content update.
3.3.4.1.1 Sending Any Error Response Message	44058 Added new section.	Y	New content added.
3.3.4.3 DFS Server Notifies SMB Server That DFS Is Active	Added new section.	Y	New content added.
3.3.4.4 DFS Server Notifies SMB Server That a Share Is a DFS Share	Added new section.	Y	New content added.
3.3.4.5 DFS Server Notifies SMB Server That a Share Is Not a DFS Share	Added new section.	Y	New content added.
3.3.5.4 Receiving an SMB_COM_TREE_CONNECT_ANDX Request	Updated the user new session key to be 16 bytes.	N	Content update.
3.3.5.4 Receiving an SMB_COM_TREE_CONNECT_ANDX Request	48291 Updated the error code used if access to the share isn't granted.	Y	Content update.
3.3.5.5 Receiving an SMB_COM_NT_CREATE_ANDX Request	Changed "TreeConnect.Share.IsDfs is TRUE" to "Server.IsDFSCapable is TRUE and Server.Share.IsDfs is TRUE".	N	Content update.
3.3.5.5 Receiving an SMB_COM_NT_CREATE_ANDX Request	48593 Updated the supported OSes in the product behavior note for the values return in the VolumeGUID and FileId fields.	N	Product behavior note updated.
3.3.5.6 Receiving an	48291 Updated the error code used if access to the share	Y	Content update.

Release: Tuesday, April 6, 2010

Section	Tracking number (if applicable) and description	Major change (Y or N)	Revision Type
SMB_COM_OPEN_ANDX Request	isn't granted.		

8 Index

Release: Tuesday, April 6, 2010