[MS-FSSO]: File Access Services System Overview

This document provides an overview of the File Access Services System Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol Family System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

File Access Services are the software and protocols that enable network file access and sharing in Windows operating environments. File Access Services allow a client computer to discover, access, and share files that are hosted on, and made available by, a server computer.

This document describes the intended functionality of the File Access Services System, how it interacts with systems and applications that need file services, and how it interacts with administrative clients to configure and manage the system. File Access Services uses multiple protocols for file access and file server administration. This document lists those protocols and describes how they are used to implement the File Access Services System.

Revision Summary

Date	Revision History	Revision Class	Comments
08/14/2009	0.1	Major	First Release.
09/25/2009	0.2	Minor	Updated the technical content.
11/06/2009	0.2.1	Editorial	Revised and edited the technical content.
12/18/2009	1.0	Major	Updated and revised the technical content.
01/29/2010	2.0	Major	Updated and revised the technical content.
03/12/2010	2.0.1	Editorial	Revised and edited the technical content.
04/23/2010	2.0.2	Editorial	Revised and edited the technical content.
06/04/2010	2.0.3	Editorial	Revised and edited the technical content.
07/16/2010	2.1	Minor	Clarified the meaning of the technical content.
08/27/2010	2.2	Minor	Clarified the meaning of the technical content.
10/08/2010	3.0	Major	Significantly changed the technical content.
11/19/2010	4.0	Major	Significantly changed the technical content.
01/07/2011	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	4.0	No change	No changes to the meaning, language, or formatting of

Date	Revision History	Revision Class	Comments
			the technical content.
03/25/2011	5.0	Major	Significantly changed the technical content.
05/06/2011	6.0	Major	Significantly changed the technical content.
06/17/2011	6.1	Minor	Clarified the meaning of the technical content.
09/23/2011	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	7.0	Major	Significantly changed the technical content.
03/30/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	7.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This Protocol Family System Document (PFSD) is primarily intended to cover the Protocol Family as a whole. In conjunction with Member Protocol Technical Documents (TDs), which are intended to cover Member Protocols, it presents the rules for information exchange relevant to those Member Protocols and the Protocol Family that are used to interoperate or communicate with a Windows operating system in its various environments.

The File Access Services System Overview describes the purpose and use of the protocols required for network File Access Services interoperation with Windows systems, in order to assist the reader in implementing Windows-compatible File Access Services clients, servers, and management tools.

It contains references to the File Access Services protocols, descriptions of use cases, an overview of how to use the protocols to implement the use cases, an abstract data model of the state information that is shared between the protocols, and descriptions at several levels of abstraction that will be helpful to file services architects and developers.

The purpose of the File Access Services System is to allow applications to access and share files located on a file server, using a network between them, in a secure and managed environment. File sharing supports the collaborative development of documents, code, or any type of file and their subsequent publication, distribution and further evolution. Centralizing file storage on file servers offers several benefits, including the following:

- Centralizes data management, including backup.
- Supports organizing data in a taxonomy (a file hierarchy) that is meaningful to a community of users, instead of requiring users to organize their own copies of the same data.
- Supports a "pull" model for document distribution, allowing users to seek out data when needed, rather than have to organize data that is "pushed" to them, for example in email.
- Saves storage and network bandwidth in email systems, by allowing users to refer to files on file servers, using hyperlinks embedded in email messages.

File Access Services may be used between any pair of computers, with one computer acting as client and the other as server. A given computer may act as a file services client, a file server, or both.

1.1 Glossary

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

access control entry (ACE)
access control list (ACL)
Active Directory
Active Directory domain
browser
browser client
browser server
client
computer name
discretionary access control list (DACL)
disk
Distributed Component Object Model (DCOM)
Distributed File System (DFS)

Distributed File System (DFS) client
DFS link
DFS namespace
DFS namespace, domain-based
DFS namespace, standalone
DFS path
DFS referral
domain
domain controller (DC)
domain master browser server
Domain Name System (DNS)
endpoint
FAT32 file system
file
file allocation table (FAT)
file system
file system control (FSCTL)
forest
Group Policy
GUID-based DNS name
handle
Lightweight Directory Access Protocol (LDAP)
named pipe
NetBIOS
NetBIOS name
NT file system (NTFS)
path
primary domain controller (PDC)
principal
remote procedure call (RPC)
security descriptor
security identifier (SID)
server
service
session
share
system access control list (SACL)
UncPath
Universal Naming Convention (UNC)
Windows Internet Name Service (WINS)

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

workgroup

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

host name
share name

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

classification rule
directory quota
file group

file management job
file screen

The following terms are defined in [MS-SMB2]:

branch cache
connection
Local object store
oplock
Oplock Break
Session

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

group identifier(group ID or GID)
SUNRPC
user identifier (user ID or UID)
XDR

The following terms are specific to this document:

Admin Client: Instance of a **client** of the **File Service** administration protocols defined in [MS-DFSNM], [MS-FSRM], [MS-RRP], or [MS-SRVS].

Browser Service: A combination of **browser servers** and **browser clients** that work together to provide the functionality defined in [MS-BRWS], along with the [MS-BRWS] supporting protocol defined in [MS-BRWSA].

DFS File Client: See [MS-GLOS] definition of **Distributed File System (DFS) client**.

DFS Service: A **service** on the **file server** that implements the **server** functionality of the Namespace Referral protocol defined in [MS-DFSC] and Namespace Management protocol defined in [MS-DFSNM].

File Access Protocol: A protocol that enables remote access to a portion of a local **Object Store**, and supports **file system** semantics. Specifically in this document this means the **SMB Access Protocols** and **NFS Access Protocols**.

File Client: Instance of an **NFS File Client**, **SMB File Client**, or **DFS File Client**.

file server: Computer hosting one or more instances of a **File Service**.

File Service: Instance of an **NFS File Service** and/or an **SMB File Service**.

NFS: Refers collectively to version 2 or version 3 of the **NFS** protocol (that is, **NFS Access Protocols**) as well as the related IETF protocols. This includes, specifically, the following standards:[RFC4506], [RFC1057], [RFC1094], [RFC1813], [RFC5531], and [RFC1833].

NFS Access Protocols: Refers collectively to **NFS** version 2 [RFC1094], **NFS** version 3 [RFC1813], and NLM/NSM [C702].

NFS File Client: A **service** that implements **client** side functionality of the **NFS Access Protocols** and exposes it to applications.

NFS File Service: A **service** on the **file server** that provides access to files using some version of the **NFS Access Protocols** in combination with related IETF protocols.

NFS Share: A **share** that is accessed through the **NFS Access Protocols**.

SMB File Client: A **service** that implements client side functionality of the **SMB Access Protocols**, and exposes it to applications.

SMB File Service: A **service** on the **file server** which provides access to files using the **SMB Access Protocols** and related protocols.

SMB Access Protocols: Refers collectively to protocols defined in [MS-CIFS], [MS-SMB], [MS-SMB2], and [MS-FSCC].

SMB share: A **share** that is accessed via the **SMB Access Protocols**.

UNC path: The location of a file in a network of computers, as specified in Universal Naming Convention (UNC) syntax. Also known as **UncPath**.

The following protocol abbreviations are used in this document:

BRWS: Common Internet File System (CIFS) Browser Protocol

CIFS: Common Internet File System (CIFS/1.0) Protocol (see [MS-CIFS])

DFSC: Distributed File System (DFS): Referral Protocol

DFSNM: Distributed File System (DFS): Namespace Management Protocol

FSRM: File Server Resource Manager Protocol

RAP: Remote Administration Protocol

RRP: Windows Remote Registry Protocol

SRVS: Server Service Remote Protocol

SMB: Server Message Block (SMB) Protocol

SMB2: Server Message Block (SMB) Version 2.0 Protocol

UNMP: User Name Mapping Protocol

WKST: Workstation Service Remote Protocol

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. Note that in [RFC2119] terms, most of these specifications should be imperative, to ensure interoperability. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

Any specification that does not explicitly use one of these terms is mandatory, exactly as if it used MUST.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

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.

[C702] The Open Group, "Protocols for Interworking: XNFS, Version 3W", C702, February 1998, http://www.opengroup.org/public/pubs/catalog/c702.htm

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, http://www.opengroup.org/public/pubs/catalog/c706.htm

[MS-ADSO] Microsoft Corporation, "Active Directory System Overview".

[MS-AUTHSO] Microsoft Corporation, "Windows Authentication Services System Overview".

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

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

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

[MS-DCOM] Microsoft Corporation, "Distributed Component Object Model (DCOM) Remote Protocol".

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

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

[MS-DISO] Microsoft Corporation, "Domain Interactions System Overview".

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

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

[MS-FSRM] Microsoft Corporation, "File Server Resource Manager Protocol".

[MS-GPSO] Microsoft Corporation, "Group Policy System Overview".

[MS-PCHC] Microsoft Corporation, "Peer Content Caching and Retrieval: Hosted Cache Protocol".

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

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

[MS-RRP] Microsoft Corporation, "Windows Remote Registry Protocol".

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Protocol Versions 2 and 3".

[MS-SRVS] Microsoft Corporation, "Server Service Remote Protocol".

[MS-UNMP] Microsoft Corporation, "User Name Mapping Protocol".

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

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, http://www.ietf.org/rfc/rfc768.txt

[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

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987, http://www.ietf.org/rfc/rfc1034.txt

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.txt

[RFC1057] Sun Microsystems, Inc., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1057, June 1998, http://www.ietf.org/rfc/rfc1057.txt

[RFC1094] Sun Microsystems, Inc., "NFS: Network File System Protocol Specification", RFC 1094, March 1989, http://www.ietf.org/rfc/rfc1094.txt

[RFC1813] Callaghan, B., Pawlowski, B., and Staubach, P., "NFS Version 3 Protocol Specification", RFC 1813, June 1995, http://www.ietf.org/rfc/rfc1813.txt

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995, http://www.ietf.org/rfc/rfc1833.txt

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

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.txt

[RFC2307] Howard, L., "An Approach for Using LDAP as a Network Information Service", RFC 2307, March 1998, http://www.ietf.org/rfc/rfc2307.txt

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006, http://www.ietf.org/rfc/rfc4506.txt

[RFC5531] Thurlow, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 5531, May 2009, http://www.rfc-editor.org/rfc/rfc5531.txt

1.2.2 Informative References

[DAVIES-TCPIP] Davies, J., "Windows Server 2008 TCP/IP Protocols and Services", Microsoft Press, January 2008, ISBN-13: 9780735624474.

[FSBO] Microsoft Corporation, "File System Behavior in the Microsoft Windows Environment", June 2008, http://download.microsoft.com/download/4/3/8/43889780-8d45-4b2e-9d3a-c696a890309f/File%20System%20Behavior%20Overview.pdf

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

[MSFT-UNIXPERM] Microsoft Corporation, "Permissions In Microsoft Services for UNIX v3.0", http://technet.microsoft.com/en-us/library/bb463216.aspx

[MSFT-DNS] Microsoft Corporation, "DNS", updated January 2005, http://technet2.microsoft.com/windowsserver/en/library/66add8fa-0348-4cc4-94d1-6d68127290881033.mspx?mfr=true

[MSFT-NTFS] Microsoft Corporation, "NTFS Technical Reference", March 2003, http://technet2.microsoft.com/WindowsServer/en/Library/81cc8a8a-bd32-4786-a849-03245d68d8e41033.mspx

[MS-PSSO] Microsoft Corporation, "Print Services System Overview".

[MS-WINSRA] Microsoft Corporation, "Windows Internet Naming Service (WINS) Replication and Autodiscovery Protocol".

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

[NBF] Microsoft Corporation, "Comparison of Windows NT Network Protocols", November 2006, http://support.microsoft.com/kb/q128233

[NIS] Sun Microsystems, Inc., "System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)", http://docs.sun.com/app/docs/doc/816-4556

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

[RFC4795] Aboba, B., Thaler, D., and Esibov, L., "Link-Local Multicast Name Resolution (LLMNR)", RFC 4795, January 2007, http://www.ietf.org/rfc/rfc4795.txt

2 Overview

Section 1, "Introduction", describes this Protocol Family System Document. This section introduces the system that is being documented.

2.1 System Summary

Most businesses and many personal computing environments require file hierarchies stored in a **file system** (referred to in this document as an **object store**) on one computer to be accessed and manipulated by applications on other computers. This scenario is typically referred to as file sharing or remote file access. A key goal is to ensure application compatibility by providing broadly the same semantics to **clients** as if the shared object store were local to them. Sharing the files through a protocol with more limited semantics, such as FTP or HTTP, would not provide this compatibility.

The File Access Services System addresses this goal with the following set of logical components:

File services, which share portions of an object store on one computer (acting as a **server**) for access and modification by other computers (acting as clients) through a **file access protocol**. Windows supports two primary file access protocol families - **NFS** and SMB. In addition to file access protocols, File Services support various management protocols. These facilitate management of the File Service, including configuration and discovery of shares. Computers hosting File Services are commonly referred to as **file servers**.

File Clients, which allow applications to access and modify content shared by a corresponding File Service, and typically provide applications with semantics broadly similar to the File Client's local object store. In addition to file access protocols, File Clients also use various management protocols for discovery of **file shares**.

A **DFS Service**, which maintains one or more hierarchal virtual namespaces, and maps different directories in a virtual namespace to one or more file shares on computers hosting File Services. This allows many file shares to be reached through a larger virtual namespace without the need to directly discover and address each file share. When doing the mapping, the proximity of the File Client to a particular File Service instance may be taken into account. The virtual namespace is made accessible to **SMB File Clients** by the **SMB File Service**, and is not accessible through the **NFS File Service**.

An **Admin Client**, which uses management protocols to configure and query the state of **services** such as the DFS Service and the File Services. The Admin Client is also used to configure various policies that apply to the object store on a file server, such as quotas (limits on user **disk** space use), screening (restrictions on the type of content allowed to be stored), and so on.

Physically, the File Access Services System consists of one or more computers each running some or all of the logical components previously mentioned. Refer to the diagram in section 5.4. It is very common for computers to be running both File Client(s) and File Service(s).

Typical usage of the File Access Services System would start with a system administrator configuring various services through an Admin Client, which are then discovered and accessed by users and applications through File Client(s). The processing model for all protocols included in the system is client-server, with the caveat that a computer acting as a server for one protocol may simultaneously be acting as a client for the same and/or different protocols.

The File Access Services System interacts with several other Windows components and systems (for a detailed list see section 3.3.2). Key interactions are as follows:

- **Browser Service** - used for discovery of computers hosting SMB-based file and print services on the network.

- Domain Interactions System - provides centralized identity and account management for users and computers.
- **Active Directory** System - persistence of some file services state, such as a **domain DFS namespace**.
- Network infrastructure components - for example, transports (such as TCP), and name resolution (such as **DNS**).
- Security - for example, NTLM and Kerberos, for **authenticating** and authorizing access to file services.
- Object Store - typically a file system, for persistence of user and application data.
- **Group Policy** - File Access Services System is used by clients to access Group Policy configuration, and Group Policy can be used to configure some aspects of file services.
- Printing subsystem - uses File Access Services for sharing of printers.

The File Access Services System exists within Windows as a set of drivers and services. This functionality is consumed by other Windows system services and applications. Users are not expected to interact directly with the File Access Services System, but rather indirectly, through text or graphical shells, applications, and higher level Windows APIs.

2.2 List of Member Protocols

The File Access Services System contains the following member protocols. See section 5.3.1 for more detail, and section 5.3.2 for protocol groupings.

Common Internet File System, as defined in [MS-CIFS]. This protocol supports the sharing of file and print resources between computers.

Security considerations are described in [MS-AUTHSO]. Further details regarding authentication and authorization are described in:

- Kerberos Protocol Extensions [MS-KILE]: This protocol specifies extensions to the Kerberos Network Authentication Service (V5) protocol [RFC4120]. These extensions provide additional capability for authorization information including group memberships, interactive logon information, and integrity levels.
- NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP]: This protocol is used by certain file access protocols for authentication between clients and servers.

Common Internet File System (CIFS) Browser Protocol as defined in [MS-BRWS] and Common Internet File System (CIFS) Browser Auxiliary Protocol as defined in [MS-BRWSA]. These protocols are used to communicate with servers that are acting as clearing houses for services available on the network. These services include printing and file sharing.

Distributed File System (DFS): Referral Protocol, as defined in [MS-DFSC]. This protocol is used by SMB File Clients to resolve **paths** in a distributed virtual namespace.

Distributed File System (DFS): Namespace Management Protocol, as defined in [MS-DFSNM]. This protocol allows remote management of DFS Services.

File System Control Codes, as defined in [MS-FSCC]. This protocol defines the network format of native Windows structures used within other protocols such as [MS-SMB].

File Server Resource Manager Protocol, as defined in [MS-FSRM]. This protocol is used to manage file system (object store) policies on a file server.

NFS: Network File System Protocol, as defined in [RFC1094] version 2 and [RFC1813] version 3. This protocol supports the sharing of file resources between computers.

Network Lock Manager (NLM) and Network Status Monitor (NSM) protocols, as defined in [C702]. These protocols are used in conjunction with the **NFS file access protocols** to provide support for file locking and service status monitoring.

Remote Administration Protocol, as defined in [MS-RAP]. This protocol is used for server discovery and remote administration. The administrative functions can use the protocol defined in [MS-BRWS] for server discovery as an alternative to that defined in [MS-RAP].

Server Message Block (SMB) Protocol, as defined in [MS-SMB]. This protocol defines extensions to the [MS-CIFS] protocol.

Server Message Block (SMB) Version 2 Protocol, as defined in [MS-SMB2]. This protocol shares and extends concepts from [MS-SMB] and [MS-CIFS], but once negotiated, is a completely new and separate command set.

Server Service Remote Protocol, as defined in [MS-SRVS]. This protocol is used for remote administration of file servers.

Workstation Service Remote Protocol, as defined in [MS-WKST]. This protocol supports remote query and configuration of certain aspects of an [MS-SMB] or [MS-SMB2] File Client.

2.3 Relevant Standards

This section provides a brief summary of standards that are relevant to the File Access Services System. See section 5.1 for a detailed discussion of all protocols used within the File Access Services System.

The following standards are transport protocols directly used by the File Access Services System:

Transmission Control Protocol (TCP), as specified in [RFC793].

User Datagram Protocol (UDP), as specified in [RFC768].

Protocol Standard for a NetBIOS Service on a TCP/UDP Transport (NetBIOS over TCP), as specified in [RFC1001] and [RFC1002].

DCE 1.1: Remote Procedure Call (XOpen RPC), as specified in [C706].

RPC: Remote Procedure Call Protocol Specification Version 2, as specified in [RFC5531].

XDR: External Data Representation Standard, as specified in [RFC4506].

The File Access Services System also uses the following standards to define **computer names**, **workgroup** names, and domain names:

Protocol Standard for a NetBIOS Service on a TCP/UDP Transport (NetBIOS names), as specified in [RFC1001] and [RFC1002]. Domain Names, as specified in [RFC1034], [RFC1035].

The Network File System (NFS) implementation within the File Access Services System makes use of:

RPC: Remote Procedure Call Protocol Specification Version 2 and Binding Protocols for ONC RPC Version 2 (Sun RPC Bind and PortMap protocols), as specified in [RFC1057], and [RFC1833].

The following standard is used directly by the File Access Services System, but is external to the system:

Lightweight Directory Access Protocol, (v3) as specified in [RFC2251] and [RFC2307].

3 Foundation

This section describes the theoretical and practical information needed to understand this document and this system.

3.1 Background Knowledge and System-Specific Concepts

This section summarizes:

- Background knowledge required to understand this document.
- Concepts that are specific to this system.

For background on the Windows System, see [MS-WSO].

3.1.1 Domains, Workgroups, and Naming

File Access Services are used within both workgroup and domain-based networks. Workgroup-based networks are designed for home and small business deployments. Domain-based networks are designed for centralized administration through a **domain controller**, and can scale to support large enterprises.

With either type of network, a computer may have a network-visible name in order for the services on the computer to be accessible. Note that for IP connectivity, it must have at least an IP address. There are two primary families of computer names supported by Windows - a Domain Name System (DNS) name, which is a hierarchical name, and a **NetBIOS** name, which is a flat name (that is, non-hierarchical). Both types of networks have the ability to logically group computers through use of a single name, referred to as the domain name or the workgroup name.

If a **DNS name** is used, it must be unique within the workgroup or the domain for correct and secure operation. Active Directory-style domains must have a DNS domain name and may optionally have a NetBIOS domain name. <1>

The NetBIOS name should be unique as well, but this is not, strictly speaking, required. If DNS and NetBIOS are deployed together, by convention the NetBIOS name is the most specific (leftmost) element of the DNS name, truncated to 15 characters. However this can lead to naming conflicts in some situations, and therefore some organizations create a NetBIOS domain name not related to the DNS name.

For additional information about DNS, see [MSFT-DNS]; for NetBIOS, see [RFC1001] and [RFC1002].

Windows File Access Services are often configured with a domain controller to provide multi-computer administration and centralized storage of account information. Domain services includes functionality such as how to create a relationship with a domain (for example, join a domain) by communicating with a domain controller, how domain state is maintained, and authentication and authorization actions.

Domain services were originally provided by a Windows NT 4.0 operating system style domain (see [MS-DISO]). This primitive style of domain was later supplanted by an Active Directory-style domain. <2> An Active Directory-style domain supports Active Directory (AD), **LDAP**, more secure authentication (Kerberos), and other advanced configurations and features.

For additional information on domain services, see [MS-DISO].

3.1.2 Identity, Accounts, and Account Types

Windows File Access Services provide secure access to files, ensuring that a user is authorized to access the content before the content is returned to the user. However, authorization can only occur after the identity of the user has been verified. Thus identity management is a core capability leveraged by Windows File Access Services.

Identity and proof of that identity is the basis for all interactions with a domain. The identity is represented by an account in the account database for the domain, and the proof is by knowledge of the password or similar secret associated with that account. By knowing both, an entity can establish an **authenticated connection**. Note that the domain controller, through use of the trusted third-party model, can be used to broker authenticated connections between two **principals**.

There are three main types of accounts. The user account is used to represent a person. The service account is very similar to a user account, but is used to represent a specific identity for an application running on the network. The computer account is used to represent a computer that participates in the domain.

Principals in the domain are responsible for knowing the name used on their accounts. For accounts representing people, the person must know the account name. For a service or computer, some configuration on the client of the domain typically indicates the name that will be used. In the same way, the person must possess the key that is used to authenticate the user account. Typically, this is a password, but may also be an asymmetric key contained on an external device such as a smart card.

For additional information, see [MS-DISO] and [MS-AUTHSO].

3.1.3 Domains and Forests

Windows File Access Services support access and namespace virtualization between different domains. Domains can be linked together in a number of ways; such a link is termed a trust. A single domain that is not linked to any other domains is the sole source of authority for all principals within that domain. Domains can be linked in several different ways. Linking domains by establishing a trust relationship between the two domains indicates that the domains trust each other to authenticate the identities of accounts within each domain. The trust need not be complete; the nature of the trust is an administrative decision made by each domain administrator.

Domains, through their DNS names, can be linked to establish trust that mirrors the hierarchy expressed in their names. For example, child1.example.com and child2.example.com would typically be configured to trust example.com, but neither would trust each other directly. Domains linked this way are termed trees. Two or more trees can be linked together to create a **forest**, where the trust is established at the root of each tree. Domain trust can be established without a forest. Any two domains can be arbitrarily linked together through a trust.

For additional information regarding domains and forests, see [MS-DISO].

3.1.4 Group Policy and Active Directory

In a domain-joined environment, Active Directory provides a central store for identity and account information as well as storage for other systems and applications. The Active Directory System also performs a key role in the Windows domain model of authentication, because it is used to store all the accounts in the domain.

Group Policy can be used by administrators to define and manage desired computer configurations or policy settings for a large number of users and computers within an Active Directory environment.

The Windows File Access Services System is designed to work with these two systems when domain-joined. See [MS-GPSO] and [MS-ADSO] for an overview of these systems.

3.1.5 Windows Branch Cache

The Windows File Access Services System is designed to leverage the Peer Content Caching and Retrieval: Hosted Cache Protocol [MS-PCHC] to reduce wide area network (WAN) bandwidth utilization for branch office deployments. A hosted cache provides the ability to cache and retrieve content securely from a local server within the branch office. The framework supports the use of a dynamically provisioned or a pre-provisioned hosted cache.

SMB version 2.1, as specified in [MS-SMB2], enables an encrypted hash list of the content to be retrieved from the file server. Applications collocated with the SMB File Client can then attempt to retrieve the actual content from the hosted cache rather than the file server. Applications can also publish content retrieved through the SMB File Client into the hosted cache.

3.1.6 Additional Network Infrastructure

The Windows File Access Services System relies upon network infrastructure to provide address resolution and a variety of transports for its protocols. Address resolution from a name to an IP address can be provided by a variety of mechanisms, but the most common are DNS [RFC1034] and [RFC1035], Link-Local Multicast Name Resolution (LLNMR) [RFC4795] for DNS names, and **Windows Internet Name Service (WINS)** [MS-WINSRA] for **NetBIOS names**. For a general overview of Windows DNS, see [MSFT-DNS].

The File Access Services System also uses a variety of transport protocols for File Access Services System protocols. See sections 4.3.2 and 5.1.

For a historical overview of Windows network protocols in the 1990s, see [NBF]. For an overview of Windows TCP/IP networking capabilities, see [DAVIES-TCPIP].

3.1.7 System-Specific Concepts

Today's file server can be deployed in a wide variety of environments, making the system appear more complex than is typically utilized by most deployments. This section will outline some of the common deployment scenarios to aid in the understanding of the File Access Services System. Deployments of a file server can be within many environments, including the home, the small/medium business, and the enterprise.

- In the home, typically many of the more advanced features are not deployed. Often, just simple service discovery (through [MS-BRWS]) and file sharing using a single access protocol (often some version of SMB) within a workgroup is used, with no centralized identity management. Often local computers have no resolvable DNS name, so all communication to the local file server is done using NetBIOS names.

- In a small/medium business, depending upon the sophistication of the company, a workgroup or a domain may be deployed. As discussed previously, the domain model enables centralized identity management, centralized policy management, and substantially improved security for network access. In addition, the **UNC pathname** may be virtualized through the **Distributed File System (DFS)**: Referral Protocol to allow file server locations to be moved transparently to the end user.

- In large enterprises, the File Access Services System can be complex. Large enterprises often deploy virtualized UNC pathnames to allow multiple file servers to serve the content in a transparent way, have heterogeneous clients, some of which prefer to communicate using the NFS Access Protocols, and some of which prefer to communicate with the **SMB Access Protocols**.

Both client and server may choose to deploy multiple protocol versions simultaneously to support interoperation with multiple implementations. Therefore, for compatibility reasons, there may be multiple versions of each protocol running within each of the previously listed environments.

Access to a file **share** can be performed through either NFS Access Protocols or SMB Access Protocols (see section 5.3.2 for identification of protocol groups). Both enable network access to an underlying object store, which is provided by the operating system. The object store is implementation-dependent. The object store determines how files are named, stored, and organized on a volume. A file system manages files and folders, and the information needed to locate and access (including authorization) these items by local users. The File Access Services System provides remote users access to the local object store. There are a variety of object stores, including **NTFS**, **FAT**, **FAT32**, and third party object stores. See [MSFT-NTFS] for an overview of NTFS, and [FSBO] for common file system behaviors.

The **Universal Naming Convention (UNC)** is a standard naming format for specifying the location of network resources in Windows environments. The convention identifies the location of a service, a specific instance of the service, and in the context of the File Access Service System, the relative pathname of the resource within the File Service to a file or directory. Within the File Access Service System, the UNC is usually rendered as "\\<server>\<share>\ <remaining path>", where a share is a specific instance of a File Service present on the target server. While a UNC pathname can be used to indicate the path to a specific physical share, it can also be used to virtualize access to shares, through the use of Distributed File System (DFS): Referral and the Distributed File System (DFS): Namespace Management Protocols.

An important result of the UNC is that the list of service instances (i.e., shares) is shared by all access protocols. Clients are responsible for resolving the access protocol necessary for communicating with the resource. This enables a pathname to be shared by clients which do not use the same file access protocol, if the File Services on the file server cooperatively offer access to the same underlying object store. For example, a single **share name** of the form "\\<server>\<share>" can be used to enable both NFS Access Protocols and SMB Access Protocols file access.

SMB was originally created with three primary uses intended - file access, printer access, and inter-process communication using **named pipes**. This specification examines the SMB Access Protocols as used for remote file access. [MS-PSSO] provides a system overview of the print system, including the use of SMB and related protocols by that system. The inter-process communication role of the SMB Access Protocols is much simpler than the file access or printer access use cases. It can be used alone as a transport, and is fully specified within the SMB Access Protocols, depending upon the version of the protocol used.

Windows File Access Services supports the ability to set **directory quotas**, file screens, and **classification rules**. These rules are enforced for all local and remote access to the object store through the NFS Access Protocols and SMB Access Protocols. The File Server Resource Manager Protocol described in [MS-FSRM] enables configuration and reporting on these settings.

3.2 System Purposes

The purpose of the File Access Services System is to allow a set of actors (people or processes) to access and share files located on a file server, using a network between them, in a secure and managed environment. The files may be distributed among a number of computers in a workgroup or domain, or centralized in one or more file server computers.

3.3 System Use Cases

3.3.1 Stakeholders and Interests Summary

The stakeholders and their associated interests for the File Access Services System are as follows:

User: The User is the person who requires remote file access in order to read, create or modify files on another computer. The User is referred to using the qualifier "SMB" or "NFS" when it is necessary to distinguish User instances. The User is external to the File Access Services System, and interacts with the File Access Services System through the Application.

Application: The Application is a program that consumes file services by means of the File Client. Applications need to create, open, read, write, and delete files on the file server. The Application is external to the File Access Services System, and interacts with the File Access Services System through the File Client.

File Client: The File Client implements client-side protocol components and consumes the file services that are offered by the file server. The File Client is referred to using the qualifier "SMB" or "NFS" when it is necessary to distinguish File Client instances. The File Client is internal to the File Access Services System.

Administrator: The Administrator is the person who administers the file server. The Administrator is interested in organizing the file namespace, setting access rights, and enforcing limits (quotas) on users' file storage. The Administrator is external to the File Access Services System, and interacts with the File Access Services System through the Admin Tool.

Admin Tool: The Admin Tool is a program that offers management functionality to the Administrator by means of the Admin Client. Typical Admin Tools are command line and graphical shells, management utilities, and graphical management programs. The Admin Tool is external to the File Access Services System and makes use of the Admin Client to accomplish its work.

Admin Client: The Admin Client is code running on the Administrator's computer. The Admin Client implements client-side protocol components and consumes the file server administration services that are offered by the file server. The Admin Client is internal to the File Access Services System.

File Service: The File Service implements server-side protocol components and the file services that are consumed by the File Client and Admin Client. The File Service is referred to using the qualifier "SMB" or "NFS" when it is necessary to distinguish File Service instances. The File Service is internal to the File Access Services System.

Print Services System: The Print Services System uses certain of the File Access Services to send print jobs (as files) to a print server. This use is described in [MS-PSSO], and is not further described here. The Print Services System is external to the File Access Services System.

Group Policy System: The Group Policy System [MS-GPSO] is dependent on the File Access Services System for download of Group Policy from the System Volume (SYSVOL) to its clients. The Group Policy System is external to the File Access Services System.

3.3.2 Supporting Actors and System Interests Summary

The File Access Services System makes use of the following external systems:

Active Directory System [MS-ADSO]: The File Access Services System stores DFS namespace information in Active Directory. Additionally, the File Access Services System co-locates its DFS Service component with Directory Services on the domain controller, and discovers the DFS Service by first locating the domain controller and then attempting to connect with the DFS Service on the same computer.

Authentication Services System [MS-AUTHSO]: Authentication Services are used for authentication.

Browser Service [MS-BRWS]: The Browser Service is considered external to the File Access Services System because it can be used by other client systems. The Browser Service is described completely in the [MS-BRWS] protocol document. The File Access Services System uses the protocols described in [MS-BRWS] and [MS-RAP] to access the Browser Service in order to discover nearby computers that may host file shares. The Browser Service optionally uses [MS-BRWSA] to query configuration information for the domains from the **domain master browser server**.

Domain Interactions System [MS-DISO]: The Domain Interactions System is used by domain-joined computers during network logon. This behavior applies to File Access Services System computers when they are domain-joined, and is described in [MS-DISO].

Group Policy System [MS-GPSO]: The Group Policy System enables an administrator to maintain standard operating environments in domains for specific groups of users and computers. As software changes and policies change over time, Group Policy can be used to update an already-deployed standard operating environment. Computers participating in the File Access Services System within a domain can be expected to participate in, and be influenced by, the Group Policy System.

Hosted Cache [MS-PCHC]: The File Access Services System optionally uses a Hosted Cache to cache file content (for example, in a branch office of a corporation). The File Client, within the File Access Services System, after initial retrieval of file content from the file server, can publish the encrypted content to a Hosted Cache. Then later, when another File Client requests the content through a secure mechanism the content can be retrieved from the Hosted Cache rather than from the file server.

Identity Store [RFC2307]: The File Access Services System optionally uses an [RFC2307]-compliant LDAP store to provide and maintain a consistent user identity mapping service to the NFS Access Protocols.

Network Information Service [NIS]: The NFS File Service consumes netgroups defined in Network Information Service [NIS] to enforce client fencing on **NFS shares**.

Object Store [FSBO]: The File Access Services System is dependent on an external Object Store for storing files and directories. <3> Some of the wire-visible behavior of File Access Services protocols is not specified by the protocols themselves, and is dependent on Object Store behavior. The common behavior of Windows file systems is specified in [FSBO].

User Name Mapping Service [MS-UNMP]: The File Access Services System uses a remote **SUNRPC** service (defined in [MS-UNMP]) to convert security identity from AUTH_SYS to Windows Accounts. The **NFS File Client** makes use of [MS-UNMP] to map from Windows Accounts to AUTH_SYS credentials which are then used in NFS requests. The NFS File Service uses [MS-UNMP] to map from AUTH_SYS credentials (from received NFS requests) to Windows Accounts.

3.3.3 Use Case Diagrams

The File Access Services System offers a variety of low-level file services operations that can be used in combination by the system's clients to accomplish their higher-level file services goals. Accordingly, both summary and detail use cases are included in this document.

Each summary use case invokes detail use cases in succession as an illustration of the use of the File Access Services System. The summary use cases are designed to show typical sequences of detail use cases, and also to highlight important features of the system.

The following table provides an overview of the summary and associated detail use cases as shown in the summary use case diagrams that follow.

Summary use cases	Detail use cases
Configure File Service -- Admin Tool	Create Share SMB -- Admin Tool Create **DFS Standalone Namespace** -- Application Create **DFS link** -- Admin Tool Configure Share Directory -- Admin Tool
Find File in Workgroup -- Application	List Computers -- Application List Shares SMB -- Application List Files SMB -- Application Open File SMB -- Application Perform File Operation SMB -- Application
Find File in Domain -- Application	List Files SMB -- Application Open File SMB -- Application Perform File Operation SMB -- Application
Communicate through Shared File -- Application	Open File SMB -- Application Open File NFS -- Application Act on Directory Change Notification SMB -- Application Perform File Operation SMB -- Application Perform File Operation NFS -- Application

Although the detail use cases are independent of each other, they are grouped by summary use case in the use case diagrams in order to aid understanding. Some of the detail use cases are used in more than one summary use case, and so they appear in more than one of the following use case diagrams.

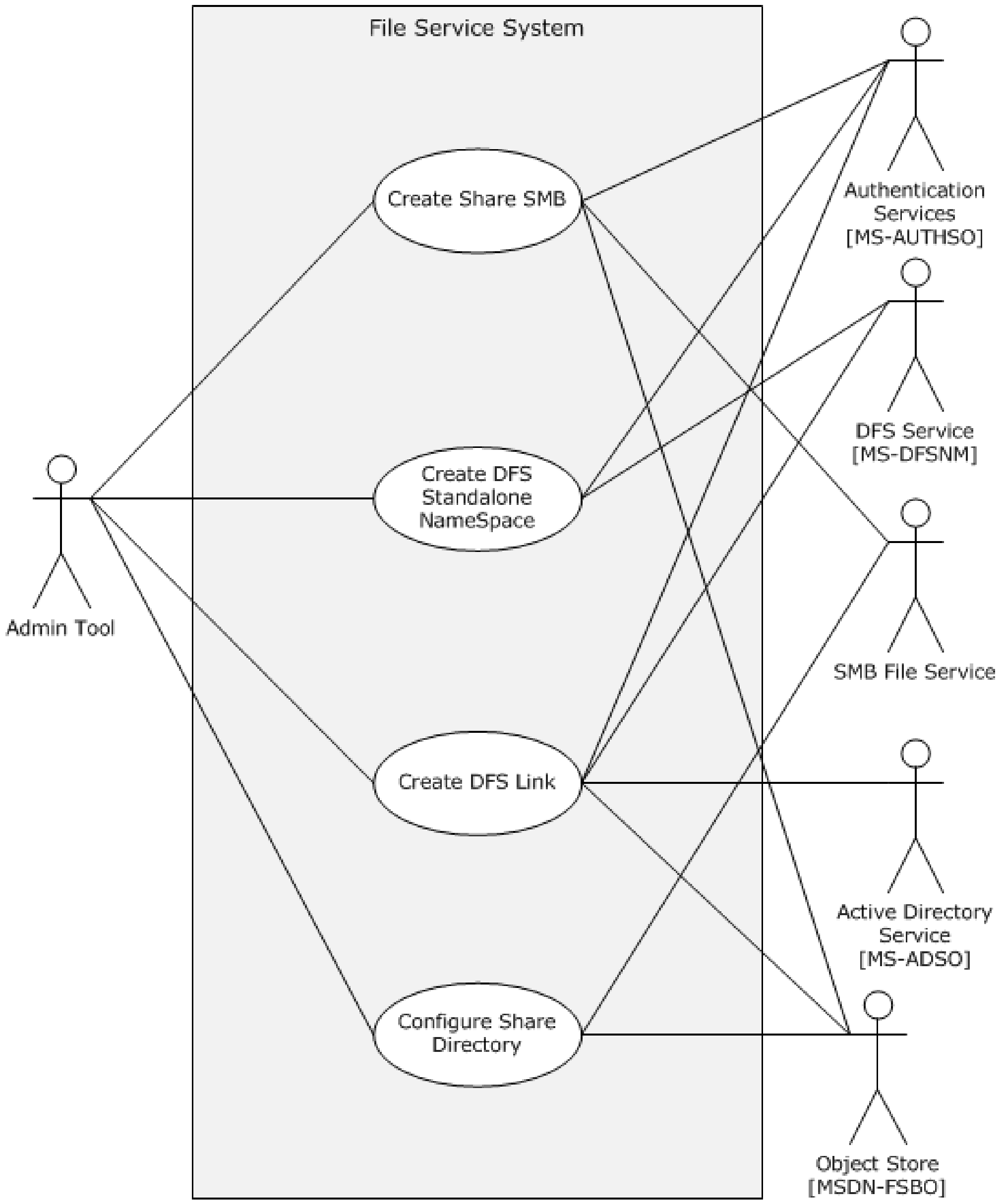

File Service System
Create Share SMB
Create DFS Standalone NameSpace
Create DFS Link
Configure Share Directory
Admin Tool
Authentication Services [MS-AUTHSO]
DFS Service [MS-DFSNM]
SMB File Service
Active Directory Service [MS-ADSO]
Object Store [MSDN-FSBO]

Figure 1: Use cases included in the Configure File Service summary use case

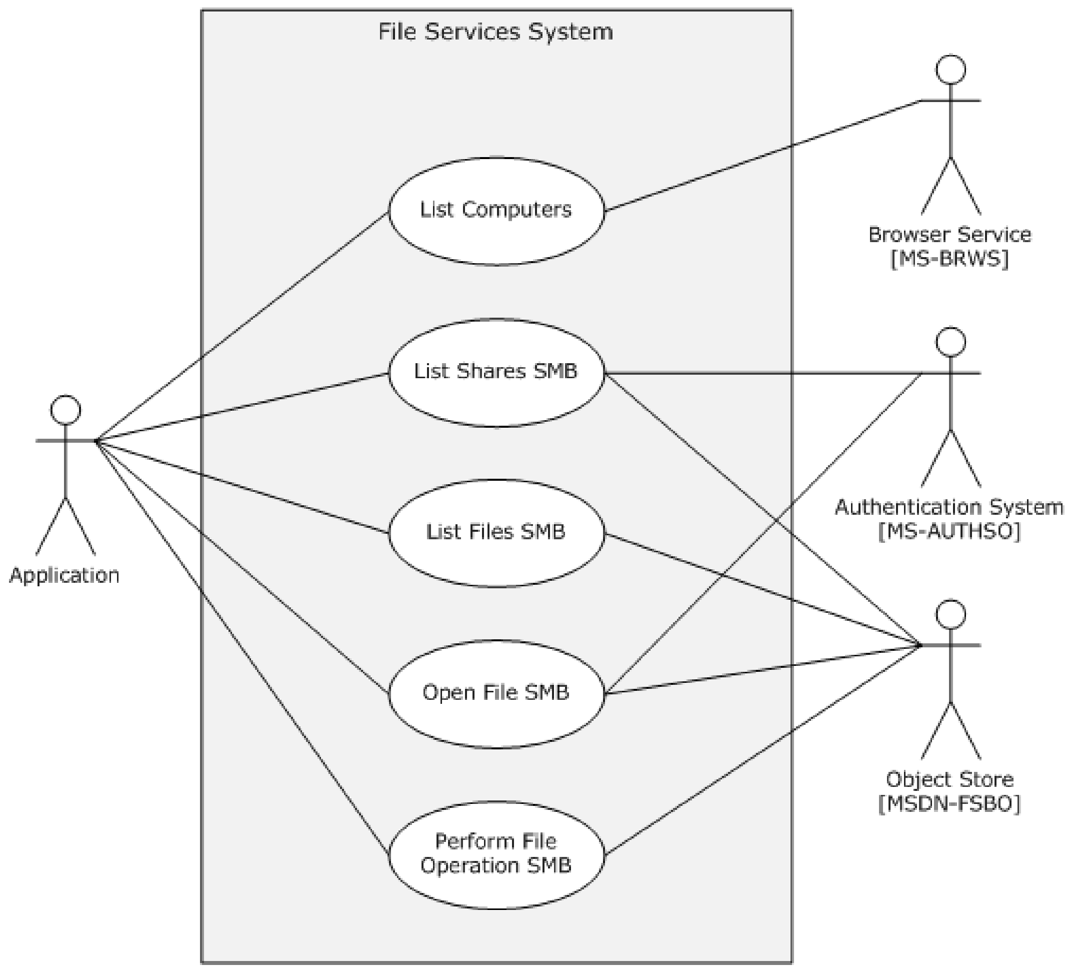


Figure 2: Use cases included in the Find File in Workgroup summary use case

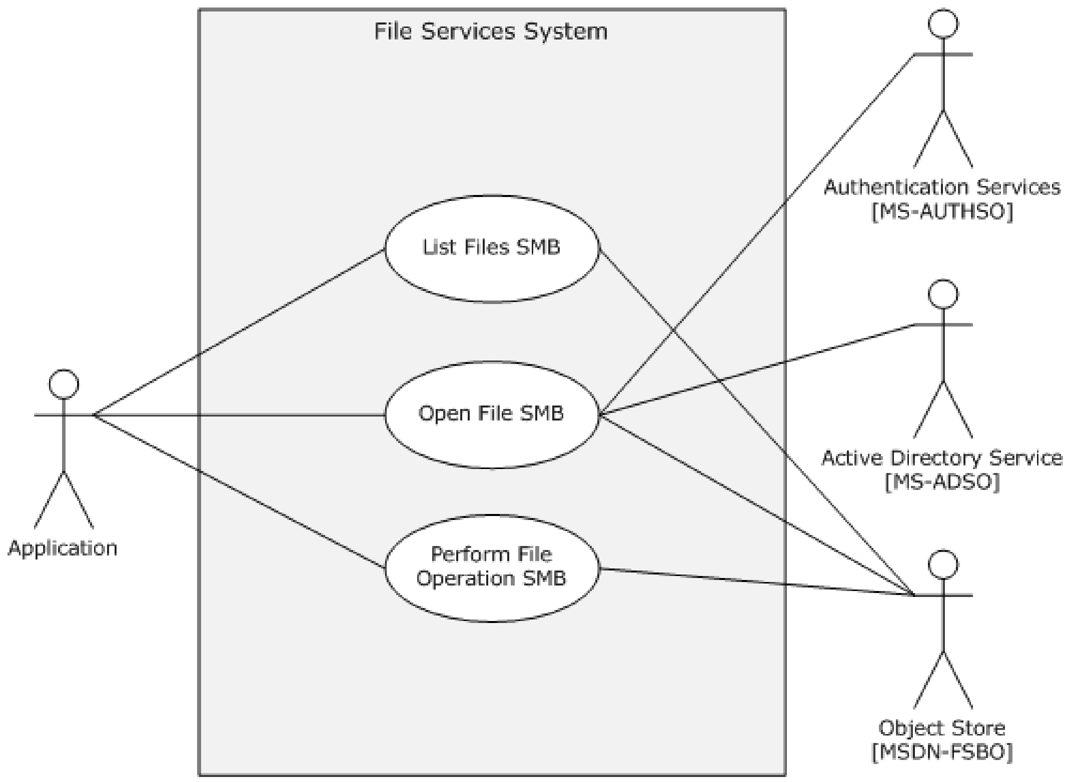

File Services System
List Files SMB
Open File SMB
Perform File Operation SMB
Application
Authentication Services [MS-AUTHSO]
Active Directory Service [MS-ADSO]
Object Store [MSDN-FSBO]

Figure 3: Use cases included in the Find File in Domain summary use case

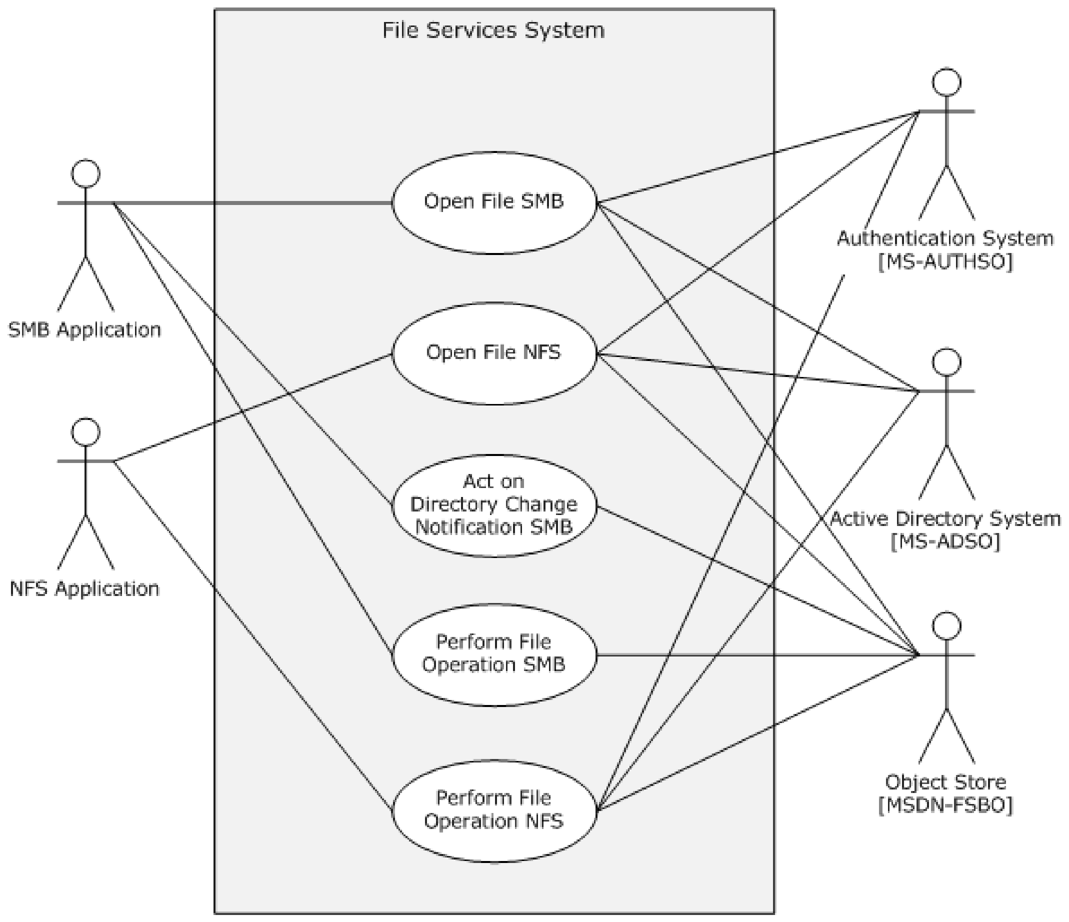

File Services System
Open File SMB
Open File NFS
Act on Directory Change Notification SMB
Perform File Operation SMB
Perform File Operation NFS
SMB Application
NFS Application
Authentication System [MS-AUTHSO]
Active Directory System [MS-ADSO]
Object Store [MSDN-FSBO]

Figure 4: Use cases included in the Communicate through Shared File summary use case

3.3.4 Use Case Descriptions

3.3.4.1 Summary Use Cases

The summary use cases illustrate Admin Tool and Application interaction with the File Access Services System. Each summary use case invokes several of the detail use cases in the order given in the Main Success Scenario. These included use case titles are underlined in the Main Success Scenarios.

3.3.4.1.1 Configure File Service - Admin Tool

See the figure "Use cases included in the Configure File Service summary use case" in section 3.3.3.

Goal: Configure File Service demonstrates typical Administrator tasks when setting up a file share by means of an Admin Tool, such as a command line utility or a graphical management program.

Context of Use: Administrator is setting up a file server, or adding a share to an existing file server.

Direct Actor: The direct actor is the Admin Tool. The Admin Tool's interest is to correctly process, execute and display the results of the commands or gestures issued by the Administrator.

Primary Actor: The primary actor is the Administrator. The Administrator's interest is expressed in administrative privileges and responsibility for using the File Access Services System to provide File Services.

Supporting Actors, Stakeholders and Interests: See included detail use case descriptions.

Preconditions: Administrator has identified a file server, a directory on the file server's Object Store to be made available as a network share, an unused **SMB share** name on the file server, and has decided on directory quota limits and **file screen**. Additionally, a desired DFS link path and target have been identified. An SMB File Service, as defined in [MS-SRVS], and a DFS Service, as defined in [MS-DFSNM], MUST be present on the file server.

Minimal Guarantees: No action is taken that affects other shares exposed by the file server. See included detail use case descriptions.

Success Guarantee: The named SMB share is configured on the file server, is provisioned as a DFS standalone namespace, contains a DFS link, and has file screen and file screen policies applied, all as directed by the invoker of this use case.

Trigger: Administrator selects a series of operations explicitly, using a command line or graphical Admin Tool, causing the Admin Tool to execute each of the following steps, in sequence.

Main Success Scenario:

1. Admin Tool creates an SMB share with the specified SMB share name and directory on the file server's Object Store using Create Share SMB.

2. Admin Tool creates a standalone DFS namespace with the specified SMB share name, using Create DFS Standalone Namespace.

3. Admin Tool creates a DFS link with the specified DFS link path and target, using Create DFS Link.

4. Admin Tool Configures directory quota and file screening on the share directory, using Configure Share Directory.

Extensions: This summary use case is illustrative of user actions, and has a nearly infinite number of possible variations. See the detail use cases for some possible extensions.

3.3.4.1.2 Find File in Workgroup - Application

See the figure "Use cases included in the Find File in Workgroup summary use case" in section 3.3.3.

Goal: Find File in Workgroup demonstrates typical User interaction with the File Access Services System, by means of a command line or graphical Application, to find and open a file of interest on a file server, in a workgroup.

Context of Use: User is required to find and open a particular file, but is unsure of where it is located within a workgroup.

Direct Actor: The direct actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by User.

Primary Actor: The primary actor is the User. The User's interest is to use the File Access Services System to access content on a file server.

Supporting Actors, Stakeholders and Interests: See included detail use case descriptions.

Preconditions: User requests to find a file server within his workgroup that advertises a purpose of interest (for example, "Group Content"), and a share on that file server with a given name (for example, "Data"). User has the name of a file to locate on this file server. User requests to explore the workgroup by means of an Application, which may be a command line or graphical user interface. The file server supports SMB Access Protocols.

Minimal Guarantees: See included detail use case descriptions.

Success Guarantee: User is able to navigate to the directory on the file server containing the file of interest, and display the file.

Trigger: User selects a series of operations explicitly, using a command line or graphical Application, causing the Application to execute each of the following steps, in sequence.

Main Success Scenario:

1. Application displays a list of computers in the workgroup and their properties, obtained by using List Computers.
2. In response to the User's selecting a computer, the Application displays a list of shares on the selected computer, obtained using List Shares SMB.
3. In response to the User's selecting a share, the Application displays a list of files and directories in the user-selected share by using List Files SMB to enumerate the files and directories, and their attributes, in the root directory.
4. In response to the User's selecting one of the displayed files, Application initiates the default open action on the user-selected file, by using Open File SMB.
5. Application uses the file **handle** to read the file contents, using Perform File Operation SMB, and renders the file contents for the User.

Extensions: This summary use case is illustrative of user actions, and has an infinite number of possible variations. See the detail use cases for some possible extensions.

3.3.4.1.3 Find File in Domain - Application

See the figure "Use cases included in the Find File in Domain summary use case" in section 3.3.3.

Goal: Find File in Domain demonstrates typical User interaction with the File Access Services System, by means of a command line or graphical Application, to find and open a file of interest on a file server, within a **domain DFS namespace**.

Context of Use: User is required to find and open a particular file, but is unsure of where it is located within a domain DFS namespace.

Direct Actor: The direct actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by the User.

Primary Actor: The primary actor is the User. The User's interest is to use the File Access Services System to access content on a file server.

Supporting Actors, Stakeholders and Interests: See included detail use case descriptions.

Preconditions: User has a UNC pathname to a domain DFS namespace, of the form \\*domain name**share name*. User requests to explore within that namespace by means of an Application, which may be a command line or graphical user interface. The DFS namespace contains a DFS link that refers to an SMB File Service.

Minimal Guarantees: See included detail use case descriptions.

Success Guarantee: User is able to navigate to the directory on the file server containing the file of interest, and display the file.

Trigger: User selects a series of operations explicitly, using a command line or graphical Application, causing Application to execute each of the following steps, in sequence.

Main Success Scenario:

1. Application displays a list of files and directories in the namespace given by the user-specified UNC path \\*domain name**share name*, by using List Files SMB to enumerate the files and directories, and their attributes, in the namespace root directory. In response to User's selecting one of the displayed directories, Application displays a list of files in the selected directory, by repeating step 1 using the UNC path \\*domain name**share name**directory-name*. This time, the Application traverses a DFS link during the Open File SMB step of List Files SMB.

2. In response to the User selecting one of the displayed files, the Application initiates the default open action on the user-selected file by using Open File SMB.

3. Application uses the file handle to read the file contents, using Perform File Operation SMB, and renders the file contents for the User.

Extensions: This summary use case is illustrative of user actions, and has an infinite number of possible variations. See the detail use cases for some possible extensions.

3.3.4.1.4 Communicate through Shared File - Application

See the figure "Use cases included in the Communicate through Shared File summary use case" in section 3.3.3.

Goal: Communicate through Shared File demonstrates how two applications can use a shared file to communicate and synchronize workflows in a request/reply fashion, while using different network file access protocols.

Context of Use: Two Applications, one using NFS Access Protocols and the other SMB Access Protocols, communicate through a shared file.

Direct Actors: There are two director actors in this use case:

- **SMB** Application: The SMB Application's interest is to correctly process, execute and display the results of the commands issued by the SMB User.

- NFS Application: The NFS Application's interest is to correctly process, execute and display the results of the commands issued by the NFS User.

Primary Actor: There are two primary actors in this use case:

- SMB User: The SMB User is a User using SMB Access Protocols provided by the SMB File Client whose interest is to access files on a file server using SMB Access Protocols.

- NFS User: The NFS User is a User using the NFS Access Protocols provided by NFS File Client whose interest is to access files on a file server using NFS Access Protocols.

Supporting Actors, Stakeholders and Interests: See included detail use case descriptions.

Preconditions: The client and server computers are configured in a domain. Two file shares, one NFS and one **SMB**, have been created on the file server, both backed by the same directory in the Object Store. Both Users have identities for use with both NFS and SMB shares. Both Users have located the desired file shares and have navigated to the desired directory in each share. The NFS application has data ready to place in a file.

Minimal Guarantees: No action is taken that affects other files exposed by the file server.

Success Guarantee: Users are able to access and modify the same file using SMB and NFS Access Protocols.

Trigger: User selects a series of operations explicitly, using a command line or graphical Application, causing the Application to execute each of the following steps, in sequence.

Main Success Scenario:

1. SMB Application prepares to receive a request message file by registering for a directory change notification on a specific directory using Open File SMB to obtain a file handle to the directory, and then using Perform File Operation SMB to request directory change notification and wait on notification.

2. NFS Application requests that a new file be created in the directory, using Open File NFS and obtains a file handle. NFS Application then uses Perform File Operation NFS to write data into the file, and finally uses Perform File Operation NFS to close the file.

3. A directory change notification is triggered to the SMB Application via the SMB File Client using Act on Directory Change Notification SMB.

4. SMB Application opens the new file and obtains a file handle using Open File SMB, reads the file's contents using Perform File Operation SMB, processes the message, appends its reply data to the end of the file using Perform File Operation SMB, and closes the file using Perform File Operation SMB.

5. NFS Application re-opens the file and obtains a file handle using Open File NFS. NFS Application reads the file contents using Perform File Operation NFS, completes its processing of the reply message, and closes the file using Perform File Operation NFS.

Extensions: This summary use case is illustrative of user actions, and has an infinite number of possible variations. See the detail use cases for some possible extensions.

3.3.4.2 Detail Use Cases

3.3.4.2.1 Create Share SMB - Admin Tool

See the figure "Use cases included in the Configure File Service summary use case" in section 3.3.3.

Goal: To create a share for access via SMB Access Protocols.

Context of Use: The Administrator is setting up a file server, or adding a share to an existing file server.

Direct Actor: The direct actor is the Admin Tool. The Admin Tool's interest is to correctly process, execute and display the results of the commands issued by Administrator.

Primary Actor: The primary actor is the Administrator. The Administrator's interest is expressed in administrative privileges and responsibility for using the File Access Services System to provide SMB File Service.

Supporting Actors: The supporting actors for this use case are as follows:

- Admin Client: Maintains a consistent access mechanism to SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The Administrator has identified a file server, an available share name on a file server, and a target directory on the file server's object store that will host the share. An SMB File Service MUST be present on the file server, as defined in [MS-SRVS].

Minimal Guarantees: No action is taken that affects other shares exposed by the file server.

Success Guarantee: The named share is created on the file server.

Trigger: The Admin Tool receives a request from the Administrator to create a share on the file server.

Main Success Scenario:

1. Admin Tool verifies the existence of the target directory in the object store and establishes a communication channel to SMB File Service, as specified in [MS-SRVS] section 2.1.
2. SMB File Service authenticates Administrator through the mechanisms as specified in [MS-AUTHSO].
3. Admin Tool contacts SMB File Service using [MS-SRVS] section 3.1.4.7 **NetrShareAdd** to create the share on the file server.
4. SMB File Service authorizes Administrator through the mechanisms as specified in [MS-SRVS] section 3.1.4.7.
5. SMB File Service creates the requested share, storing configuration information in an implementation-specific manner.

Extensions:

1. The communication channel for [MS-SRVS] cannot be established, or it becomes disconnected:
   - Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure. Depending on when the connection failed, the share may or may not have been created.
2. User authentication fails:
   - The use case ends with failure.
3. User authorization fails:
   - The use case ends with failure.

3.3.4.2.2 Configure Share Directory - Admin Tool

See the figure "Use cases included in the Configure File Service summary use case" in section 3.3.3.

Goal: To configure file server share directory quota and a file screen.

Context of Use: The Administrator is setting up a file server, or adding a share to an existing file server.

Direct Actor: The director actor is the Admin Tool. The Admin Tool's interest is to correctly process, execute and display the results of the commands issued by the Administrator.

Primary Actor: The primary actor is the Administrator. The Administrator's interest is expressed in administrative privileges and responsibility for using the File Access Services System to provide File Service.

Supporting Actors: The supporting actors for this use case are as follows:

- Admin Client: Maintains a consistent access mechanism to the SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service by enforcing directory quotas and file screens.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The Administrator has identified a file server and a directory on its object store, and knows what directory quota and file screen to implement.

Minimal Guarantees: No action is taken that affects other directories or shares on the file server.

Success Guarantee: The requested quota limits and file screens are instantiated on the file server.

Trigger: The Admin Tool receives a request from the Administrator to configure quota and screening.

Main Success Scenario:

1. Admin Tool verifies the existence of the directory in the object store and establishes a communication channel to file server Resource Manager (a component of the File Service), as specified in [MS-FSRM] section 3.1.3.
2. Admin Tool creates a quota on the file server, as specified in [MS-FSRM] section 3.2.4.2.18.3. The created quota is set to the desired quota limit as specified in [MS-FSRM] sections 3.2.4.2.14.3 and 3.2.4.2.10.5.
3. Admin Tool creates a file screen on the file server, as specified in [MS-FSRM] section 3.2.4.2.29.3, and enumerates available file groups on the file server, as specified in [MS-FSRM] section 3.2.4.2.25.3. Blocked **file groups** of the created file screen are then updated by adding one or more of the enumerated file groups, as specified by the caller, using the **IFsrmFileScreenBase::PutBlockedFileGroups** method ([MS-FSRM] sections 3.2.4.2.26.2 and 3.2.4.2.10.5).

Extensions:

1. The communication channel for [MS-FSRM] cannot be established, or it becomes disconnected:

- Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure. Depending on when the connection failed, the file quota and file screen may or may not have been created.

3.3.4.2.3 Create DFS Standalone Namespace - Admin Tool

See the figure "Use cases included in the Configure File Service summary use case" in section 3.3.3.

Goal: To create a standalone DFS namespace for access via SMB Access Protocols, with [MS-DFSC] extensions.

Context of Use: The Administrator is setting up a file server, or adding a namespace to an existing file server.

Direct Actor: The director actor is the Admin Tool. The Admin Tool's interest is to correctly process, execute and display the results of the commands issued by the Administrator.

Primary Actor: The primary actor is the Administrator. The Administrator's interest is expressed in administrative privileges and responsibility for using the File Access Services System to provide SMB File Service.

Supporting Actors: The supporting actors in this use case are as follows:

- Admin Client: Maintains a consistent access mechanism to the SMB File Service.
- DFS Service: Permits configuration and management of DFS namespaces.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The Administrator has identified an SMB File Service and an existing SMB file share on the File Service that will be promoted to a DFS namespace. A DFS service MUST be present on the File Service, as defined in [MS-DFSNM].

Minimal Guarantees: No action is taken that affects other shares exposed by the file server.

Success Guarantee: The named share is promoted to a DFS namespace on SMB File Service.

Trigger: Admin Tool receives a request from Administrator to create a standalone DFS namespace on SMB File Service.

Main Success Scenario:

1. Admin Tool establishes a communication channel to DFS Service, as specified in [MS-DFSNM] section 2.1.
2. DFS Service authenticates Administrator through the mechanisms described in [MS-AUTHSO].
3. Admin Tool contacts DFS Service using [MS-DFSNM] section 3.1.4.1.9 **NetrDfsAddRootTarget** or [MS-DFSNM] section 3.1.4.4.1 **NetrDfsAddStdRoot** to promote the share to a namespace on the file server.
4. DFS Service authorizes Administrator through the mechanisms of [MS-DFSNM] section 3.1.4.1.9 **NetrDfsAddRootTarget** or [MS-DFSNM] section 3.1.4.4.1 **NetrDfsAddStdRoot**, as appropriate to the call.

5. DFS Service performs the action.

Extensions:

1. The communication channel for [MS-DFSNM] cannot be established, or it becomes disconnected:
   - Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure. Depending on when the connection failed, the DFS standalone namespace may or may not have been created.
2. User authentication fails:
   - The use case ends with failure.
3. User authorization fails:
   - The use case ends with failure.

3.3.4.2.4 Create DFS Domain Namespace - Admin Tool

Goal: To create a domain DFS namespace for access via SMB Access Protocols with [MS-DFSC] extensions.

Context of Use: The Administrator is setting up a file server, or adding a namespace to an existing file server.

Direct Actor: The direct actor is the Admin Tool. The Admin Tool's interest is to correctly process, execute and display the results of the commands issued by the Administrator.

Primary Actor: The primary actor is the Administrator. The Administrator's interest is expressed in administrative privileges and responsibility for using the File Access Services System to provide SMB File Service.

Supporting Actors: The supporting actors in this use case are as follows:

- Admin Client: Maintains a consistent access mechanism to the SMB File Service.
- DFS Service: Permits configuration and management of DFS namespaces.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The Administrator has identified an SMB file service and an existing SMB file share on the file service that will be promoted to a DFS domain namespace. A DFS Service MUST be present on the SMB File Service, as defined in [MS-DFSNM].

Minimal Guarantees: No action is taken that affects other shares exposed by the file server or other domain DFS namespaces defined in the domain of the SMB File Service.

Success Guarantee: The named share is promoted to a DFS namespace on the SMB File Service with corresponding metadata written to the Active Directory System.

Trigger: The Admin Tool receives a request from the Administrator to create a DFS namespace on the SMB File Service.

Main Success Scenario:

1. Admin Tool establishes a communication channel to the DFS Service, as specified in [MS-DFSNM] section 2.1.

2. DFS Service authenticates the Administrator through the mechanisms as specified in [MS-AUTHSO].

3. Admin Tool contacts DFS Service using [MS-DFSNM] section 3.1.4.1.9 **NetrDfsAddRootTarget** or [MS-DFSNM] section 3.1.4.3.1 NetrDfsAddFtRoot to promote the share to a namespace on the SMB File Service.

4. DFS Service authorizes Administrator through the mechanisms of [MS-DFSNM] section 3.1.4.1.9 **NetrDfsAddRootTarget** or [MS-DFSNM] section 3.1.4.3.1 NetrDfsAddFtRoot, as appropriate to the call.

Extensions:

1. The communication channel for [MS-DFSNM] cannot be established, or it becomes disconnected:

   - Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure. Depending on when the connection failed, the namespace may or may not have been created.

2. User authentication fails:

   - The use case ends with failure.

3. User authorization fails:

   - The use case ends with failure.

3.3.4.2.5 Create DFS Link - Admin Tool

See the figure "Use cases included in the Configure File Service summary use case" in section 3.3.3.

Goal: To create a DFS link for access via SMB Access Protocols with [MS-DFSC] extensions.

Context of Use: The Administrator is setting up a file server, or maintaining a namespace on an existing file server.

Direct Actor: The director actor is the Admin Tool. The Admin Tool's interest is to correctly process, execute and display the results of the commands issued by the Administrator.

Primary Actor: The primary actor is the Administrator. The Administrator's interest is expressed in administrative privileges and responsibility for using the File Access Services System to provide SMB File Service.

Supporting Actors: The supporting actors in this use case are as follows:

- Admin Client: Maintains a consistent access mechanism to the SMB File Service.
- DFS Service: Permits configuration and management of DFS namespaces.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.

- Active Directory System [MS-ADSO]: Stores metadata related to the domain DFS namespace.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The Administrator has identified an SMB File Service hosting an instance of the given namespace, the SMB share on the SMB File Service that hosts the given namespace, the path within the share at which the link should be created, and the target that the link should refer to. A DFS Service MUST be present on the SMB File Service, as defined in [MS-DFSNM].

Minimal Guarantees: No action is taken that affects other shares exposed by the file server, other links hosted within the given namespace, or other domain DFS namespaces defined in the domain of the SMB File Service.

Success Guarantee: The specified DFS link is created within the given DFS namespace on the SMB File Service, with corresponding metadata written to the Active Directory System in the case of a domain DFS namespace.

Trigger: The Admin Tool receives a request from the Administrator to create a DFS link on the SMB File Service.

Main Success Scenario:

1. Admin Tool establishes a communication channel to DFS Service, as specified in [MS-DFSNM] section 2.1.
2. DFS Service authenticates Administrator through the mechanisms as specified in [MS-AUTHSO].
3. Admin Tool contacts the DFS Service using [MS-DFSNM] section 3.1.4.1.3 NetrDfsAdd to create the link within the namespace, also creating DFS link object in the local object store.
4. DFS Service authorizes Administrator through the mechanisms specified in [MS-DFSNM] section 3.1.4.1.3 NetrDfsAdd.
5. DFS Service performs the action.

Extensions:

1. The communication channel for [MS-DFSNM] cannot be established, or it becomes disconnected:
   - Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure. Depending on when the connection failed, the link may or may not have been created.
2. User authentication fails:
   1. The use case ends with failure.
   2. In the case of a domain DFS namespace:
      - DFS Service additionally interacts with the [MS-ADSO] Active Directory (AD) System to store metadata changes related to the DFS link, as specified in [MS-DFSNM].
3. User authorization fails:
   - The use case ends with failure.

3.3.4.2.6 List Computers -- Application

See the figure "Use cases included in the Find File in Workgroup summary use case" in section 3.3.3.

Goal: To list computers in a workgroup or domain in order to discover computers that may provide file service.

Context of Use: The User requests network file servers that can be accessed for the File Service. For browsing to span subnetworks, the User's computer MUST be a member of a domain, and the Administrator MUST enable the browser service on the **primary domain controller (PDC)** server.

Direct Actor: The direct actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by the User.

Primary Actor: The primary actor is the User. The User's interest is to use the File Access Services System to access File Services.

Supporting Actors: The supporting actors are as follows:

- File Client: Maintains a consistent access mechanism to the File Service.
- Browser Service [MS-BRWS] and [MS-BRWSA]: Maintains a list of servers in the workgroup or domain. Browser Service includes the Master Browser, Domain Master Browser, and Backup Browser computers.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: None.

Minimal Guarantees: User's own computer is returned in the list of computers. If the browser service is not running in the workgroup or domain, User's computer may be the only computer listed.

Success Guarantee: A list of servers in the workgroup or domain is returned to User.

Trigger: Application receives a request from User to display a list of computers in User's workgroup or domain.

Main Success Scenario:

1. The Application contacts the local Browser Service Master Browser using the GetBackupListRequest frame ([MS-BRWS] section 3.1.5.1.1) to retrieve the list of Backup Browsers.
2. The Master Browser performs the action.
3. When the Application receives a GetBackupListResponse frame (as specified in [MS-BRWS] section 3.1.5.1.2), it then selects and establishes a communication channel to a Backup Browser, as specified in [MS-RAP] section 2.1.
4. The Application contacts the Backup Browser using [MS-RAP] section 3.1.4.12 NetServerEnum2 or [MS-RAP] section 3.1.4.15 NetServerEnum3 to retrieve the list of available servers.
5. The Backup Browser performs the action and returns the results to the Application (as described in [MS-BRWS] section 3.3.5.6).
6. The Application displays a list of computers.

Extensions:

1. No response is received from the Master Browser:
   - Application may attempt to contact Master Browser multiple times; ultimately, the use case ends with failure.
2. The communication channel for [MS-RAP] cannot be established, or it becomes disconnected:
   - Admin Tool may attempt to establish connection multiple times with this or with an alternate Backup Browser; ultimately, the use case ends with failure.

3.3.4.2.7 List Shares SMB - Application

See the figure "Use cases included in the Find File in Workgroup summary use case" in section 3.3.3.

Goal: To list shares on a file server that are accessible via SMB Access Protocols.

Context of Use: The User has located and selected a file server, and requests any file shares on that server.

Direct Actor: The director actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by the User.

Primary Actor: The primary actor is the User. The User's interest is to discover shares on the file server.

Supporting Actors: The supporting actors are as follows:

- File Client: Maintains a consistent access mechanism to the SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The User has identified a file server of interest. An SMB File Service MUST be present on the file server, as defined in [MS-SRVS] section 3.1.

Minimal Guarantees: No action is taken that affects shares exposed by the file server.

Success Guarantee: The list of shares hosted by the file server is returned to the User.

Trigger: The Application receives a request from the User to retrieve a list of shares hosted by the file server.

Main Success Scenario:

1. Application establishes a communication channel to the SMB File Service, as specified in [MS-SRVS] section 2.1.
2. SMB File Service authenticates User through the mechanisms as specified in [MS-AUTHSO].

3. Application contacts the SMB File Service using [MS-SRVS] section 3.1.4.8 **NetrShareEnum** to retrieve the list of shares.

4. SMB File Service authorizes User through the procedure specified in [MS-SRVS] 3.1.4.8.

5. SMB File Service performs the action and returns the results to the Application as described in [MS-SRVS] section 3.1.4.8.

6. Application displays a list of file shares.

Extensions:

1. The communication channel for [MS-SRVS] cannot be established, or it becomes disconnected:
   - Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure.

2. User authentication fails:
   - The use case ends with failure.

3. User authorization fails:
   - The use case ends with failure.

3.3.4.2.8 List Shares NFS - Application

Goal: To list shares on a file server that are accessible via NFS Access Protocols.

Context of Use: The User has located and selected a file server, and requests any file shares on that server that are accessible via NFS Access Protocols.

Direct Actor: The director actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by the User.

Primary Actor: The primary actor is the User. The User's interest is to discover NFS protocol shares on the file server.

Supporting Actors: The supporting actors are as follows:

- NFS File Client: Maintains a consistent access mechanism to the NFS File Service.
- NFS File Service: Provides and maintains a secure and consistent file service.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The User has identified a file server of interest. An NFS File Service must be present on the file server, as defined in [RFC1094].

Minimal Guarantees: There may not be any NFS shares on the target file server. In this case, the Application will not list any shares. No action is taken that affects shares exposed by the file server.

Success Guarantee: A list of shares hosted by the NFS File Service is returned to the User.

Trigger: Application receives request from User to retrieve a list of shares hosted by the NFS File Service.

Main Success Scenario:

1. The Application contacts the NFS File Service using a MOUNTPROC_DUMP request as described in [RFC1094] section A.5 or a MOUNTPROC3_DUMP request as described in [RFC1813] section 5.2.2.

2. The Application displays a list of NFS file shares.

Extensions:

1. RPC Timeout Occurs:
    - Application displays an error message and the use case aborts.
2. RPC error code is returned from the NFS File Service:
    - Application displays an error message corresponding to the **RPC** Error code returned and the use case aborts.

3.3.4.2.9 List Files SMB - Application

See the figures "Use cases included in the Find File in Workgroup summary use case" and " Use cases included in the Find File in Domain summary use case" in section 3.3.3.

Goal: To list files in a network share directory that are accessible via SMB Access Protocols.

Context of Use: The User has located an SMB share, and requests to list the contents of (enumerate) its directory.

Direct Actor: The direct actor is the Application. The Application's interest is to correctly process, execute and display the results of commands issued by the User.

Primary Actor: The primary actor is the User. The User's interest is to enumerate files in a share directory.

Supporting Actors: The supporting actors are as follows:

- File Client: Maintains a consistent access mechanism to the SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The User knows the path name of the directory to enumerate.

Minimal Guarantees: Listing a directory has no effect on the state of the File Service or Object Store. There may not be any files or directories in the target directory. In this case, the Application will not list any directory contents.

Success Guarantee: The Application obtains a list of files and directories in the target directory.

Trigger: The Application receives a request from the User to display a list of files and directories in a given network share directory.

Main Success Scenario:

1. Application invokes use case Open File SMB, on the specified file path, and receives a handle to the network share directory.
2. Application contacts the SMB File Service and uses the mechanisms as defined in [MS-CIFS] section 3.2.4.27, [MS-SMB] section 3.2.4.6 or [MS-SMB2] section 3.2.4.17 to enumerate the directory.
3. SMB File Service performs the action and returns the results to the Application in the form corresponding to the mechanism requested by the application in step 2.
4. Application displays the result.

Extensions:

1. The communication channel cannot be established, or it becomes disconnected:

- Application may attempt to establish connection multiple times; ultimately, the use case ends with failure.

3.3.4.2.10 List Files NFS - Application

Goal: To list files in a network share directory that are accessible via NFS Access Protocols.

Context of Use: The User has located an NFS share and requests to list the contents of (enumerate) its directory.

Direct Actor: The director actor is the Application. The Application's interest is to correctly process, execute and display the results of commands issued by the User.

Primary Actor: The primary actor is the User. The User's interest is to enumerate files in a share directory.

Supporting Actors: The supporting actors are as follows:

- NFS File Client: Maintains a consistent access mechanism to NFS File Service.
- NFS File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.
- Identity Store: Provides and maintains a consistent user identity mapping service to the NFS File Service.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: User knows the path name of the directory to enumerate.

Minimal Guarantees: Listing a directory has no effect on the state of the NFS File Service or Object Store. There may not be any files or directories in the target directory. In this case, the Application will not list any directory contents.

Success Guarantee: The Application obtains a list of files and directories in the target directory.

Trigger: The Application receives a request from the User to display a list of files and directories in a given NFS share directory.

Main Success Scenario:

1. Application resolves the target directory using a set of LOOKUP requests as described in [RFC1094] section 2.2.5 or [RFC1813] section 3.3.3 to resolve each component of the path.
2. Application enumerates the directory using one or more READDIR requests as described in [RFC1094] section 2.2.17 or [RFC1813] section 3.3.16.
3. Application displays a list of files and directories in the NFS share.

Extensions:

1. RPC Timeout Occurs:
   - Application displays an error message and the use case aborts.
2. RPC Authentication Fails:
   - Application displays an error message and the use case aborts.
3. A protocol error code is returned from the NFS File Service:
   - Application displays an error message corresponding to the RPC Error code returned and the use case aborts.

3.3.4.2.11 Open File SMB - Application

See the figures "Use cases included in the Find File in Workgroup summary use case", "Use cases included in the Find File in Domain summary use case", and "Use cases included in the Communicate through Shared File summary use case" in section 3.3.3.

Goal: To open or create a file in a directory located in an SMB file share, optionally through an [MS-DFSC] mediated namespace.

Context of Use: The User has located a directory in an SMB file share and requests to open or create a file in that directory.

Direct Actor: The direct actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by User.

Primary Actor: The primary actor is the User. The User's interest is to open or create a file in a network share directory on an SMB File Service.

Supporting Actors: The supporting actors are as follows:

- SMB File Client: Maintains a consistent access mechanism to the SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides client authentication.
- Object Store: Stores files and directories.

Stakeholders and Interests:

- Group Policy [MS-GPSO]: Policies applicable to the object store are followed.

Preconditions: A file share has been created on the SMB File Service. The User has located the path to the file on the file share and has determined the desired open type (for example, create a new file, open existing file).

Minimal Guarantees: No action is taken that affects other files exposed by the file server as a result of this operation.

Success Guarantee: The Application obtains a handle to the file.

Trigger: Based on User interaction, the Application determines to open or create a file in a target directory.

Main Success Scenario:

1. Application establishes a communication channel to SMB File Service.
2. SMB File Service authenticates User through the mechanisms as specified in [MS-AUTHSO].
3. Depending on the protocol being used, Application directs the SMB File Client to send one of the following requests to SMB File Service:
    1. [MS-CIFS]: An SMB_COM_NT_CREATE_ANDX request as specified in [MS-CIFS] section 2.2.4.64, with an appropriate CreateDisposition as specified in the same section.
    2. [MS-SMB]: An SMB_COM_NT_CREATE_ANDX request as specified in [MS-SMB] section 2.2.4.9.1, with an appropriate CreateDisposition as specified in the same section.
    3. [MS-SMB2]: A CREATE request as specified in [MS-SMB2] section 2.2.13, with an appropriate CreateDisposition as specified in the same section.
4. SMB File Service authorizes User using the mechanisms of [MS-SMB2] 3.3.5.9, [MS-SMB] 3.3.5.5, or [MS-CIFS] 2.2.4.64 as appropriate to the request.
5. If the target of the file create resides in a DFS namespace, and the SMB File Client has indicated that it includes a mechanism to detect and traverse DFS namespaces, then the SMB File Service and SMB File Client perform additional **DFS** processing (see [MS-DFSC] section 3.1.4.1). If the SMB File Client has not indicated that it includes this mechanism, then the SMB File Service fails the operation.
6. SMB File Service performs the create operation, and returns a file handle to Application.

Extensions:

1. The communication channel cannot be established, or it becomes disconnected:
    - Application may attempt to establish connection multiple times; ultimately, the use case ends with failure.
2. User authentication fails:
    - The use case ends with failure.
3. User authorization fails:
    - The use case ends with failure.

3.3.4.2.12 Open File NFS - Application

See the figure "Use cases included in the Communicate through Shared File use case" in section 3.3.3.

Goal: To open or create a file in an NFS network share directory.

Context of Use: The User has located a directory in a network share and requests to open or create a file in that directory.

Direct Actor: The direct actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by User.

Primary Actor: The primary actor is the User. The User's interest is to open or create a file in a network share directory on an NFS File Service.

Supporting Actors: The supporting actors are as follows:

- NFS File Client: Provides and maintains a consistent access mechanism to the NFS File Service.
- NFS File Service: Provides and maintains a secure and consistent file service.
- Authentication Services [MS-AUTHSO]: Provides and maintains secure authentication services to actors providing or accessing service.
- Identity Store: Provides and maintains a consistent user identity mapping service to the NFS File Service.
- Object Store: Stores files and directories.

Stakeholders and Interests:

- Group Policy [MS-GPSO]: Policies applicable to the object store are followed.

Preconditions: The client and server computers are configured in a domain. An NFS file share has been created on the file server. The User has located the path to the file on the NFS file share and has determined the desired open type (for example, create a new file or open existing file).

Minimal Guarantees: No action is taken that affects other files exposed by the file server as a result of this operation.

Success Guarantee: The User obtains a handle to the file.

Trigger: Based on User interaction, the Application determines to open or create a file in a target directory.

Main Success Scenario:

1. Application resolves the target directory using a set of LOOKUP requests as described in [RFC1094] section 2.2.5 or [RFC1813] section 3.3.3 to resolve each component of the path.
2. Application uses one of the following options depending upon the requested operation type:
    1. (Create) Creation of the new file using the CREATE request as described in [RFC1094] section 2.2.10 or [RFC1813] section 3.3.8.
    2. (Open) Resolution of the file name (final path component) using the LOOKUP request as described in [RFC1094] section 2.2.5 or [RFC1813] section 3.3.3.

Each operation returns a file handle to the Application.

3. Application retains the file handle for further use.

Extensions:

1. RPC Timeout Occurs:
   - Application displays an error message and the use case aborts.
2. RPC Authentication Fails:
   - Application displays an error message and the use case aborts.
3. A protocol error code is returned from the NFS File Service:
   - Application displays an error message corresponding to the RPC Error code returned and the use case aborts.

3.3.4.2.13 Perform File Operation SMB - Application

See the figures "Use cases included in the Find File in Workgroup summary use case", "Use cases included in the Find File in Domain summary use case", and "Use cases included in the Communicate through Shared File summary use case" in section 3.3.3.

Goal: To perform a file operation (such as Read, Write or Delete) on a file in an SMB share directory.

Context of Use: The Application has obtained a handle to a file in an SMB share directory, and requests to perform an operation on the file.

Direct Actor: The director actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by User.

Primary Actor: The primary actor is the User. The User's interest is to access files on an SMB File Service.

Supporting Actors: The supporting actors are as follows:

- SMB File Client: Maintains a consistent access mechanism to the SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Object Store: Stores files and directories.

Stakeholders and Interests:

- Group Policy [MS-GPSO]: Policies applicable to the object store are followed.

Preconditions: A file share has been created on the SMB File Service. The User has successfully opened and obtained a handle to a file on the file share. Data for the operation is supplied by the User if applicable.

Minimal Guarantees: No action is taken that affects other files exposed by the file server as a result of this operation.

Success Guarantee: The User is notified of the status of the operation performed on the file.

Trigger: The Application receives a request from the User to perform a file operation on a file for which the Application has a file handle.

Main Success Scenario:

1. Application identifies the communication channel to the SMB File Service for SMB Access Protocols from the file handle corresponding to User's file.
2. Depending on the protocol being used, the Application directs the File Client to perform the requested operation as described in:
    1. [MS-CIFS] section 3.2.4.
    2. [MS-SMB] section 3.2.4.
    3. [MS-SMB2] section 3.2.4.
3. SMB File Service performs the requested action and returns the result through the SMB File Client to the Application in the format that corresponds to the operation requested in step 2.

Extensions:

1. The communication channel cannot be established, or it becomes disconnected:
    - Admin Tool may attempt to establish connection multiple times; ultimately, the use case ends with failure. Depending on when the connection failed, the share may or may not have been created.
2. If the requested operation is a read, and the SMB share being accessed supports hash generation (see [MS-SMB2] section 2.2.10):
    - The SMB File Client may take additional steps to retrieve data from, or publish data to, the Hosted Cache.
3. User authentication or authorization fails:
    - The use case ends with failure.

3.3.4.2.14 Perform File Operation NFS - Application

See the figure "Use cases included in the Communicate through Shared File summary use case" in section 3.3.3.

Goal: To perform a file operation (such as Read, Write or Delete) on a file in an NFS network share directory.

Context of Use: The Application has obtained a handle to a file in a network share directory, and requests to perform an operation on the file.

Direct Actor: The director actor is the Application. The Application's interest is to correctly process, execute and display the results of the commands issued by User.

Primary Actor: The primary actor is the User. The User's interest is to access files on an NFS file server.

Supporting Actors: The supporting actors are as follows:

- NFS File Client: Provides and maintains a consistent access mechanism to the NFS File Service.

- NFS File Service: Provides and maintains a secure and consistent file service.
- Authentication System [MS-AUTHSO]: Provides and maintains secure authentication services to actors providing or accessing service.
- Identity Store: Provides and maintains a consistent user identity mapping service to the NFS File Service.
- Object Store: Stores files and directories.

Stakeholders and Interests:

- Group Policy [MS-GPSO]: Policies applicable to the object store are followed.

Preconditions: The client and server computers are configured in a domain. A file share has been created on the file server. The User has successfully opened and obtained a handle to a file on the file share. Data for the operation is supplied by the User if applicable.

Minimal Guarantees: No action is taken that affects other files exposed by the file server as a result of this operation.

Success Guarantee: The User is notified of the status of the operation performed on the file.

Trigger: The Application receives a request from the User to perform a file operation on a file for which the Application has a file handle.

Main Success Scenario:

1. Application uses the file handle to perform the requested operation as described in [RFC1094] section 2.2 or [RFC1813] section 3.3.
2. The NFS File Service performs the requested action and returns the result to NFS File Client.
3. NFS File Client returns the result to Application as described in [RFC1094] section 2.2 or [RFC1813] section 3.3.

Extensions:

1. RPC Timeout Occurs:
   - Application displays an error message and the use case aborts.
2. RPC Authentication Fails:
   - Application displays an error message and the use case aborts.
3. A protocol error code is returned from the NFS File Service:
   - Application displays an error message corresponding to the RPC Error code returned and the use case aborts.

3.3.4.2.15 Act on Directory Change Notification SMB - Application

See the figure "Use cases included in the Communicate through Shared File summary use case" in section 3.3.3.

Goal: To act on a directory change notification previously requested by the Application.

Context of Use: User has obtained a handle to a directory in an SMB share and used it to monitor for changes in that directory.

Direct Actor: The direct actor is the Application (for example, command line or graphical shell).

Primary Actor: The primary actor is the User. The User's interest is to use the SMB Access Protocols provided by the SMB File Client to monitor a directory on a file server.

Supporting Actors: The supporting actors are as follows:

- SMB File Client: Provides and maintains a consistent access mechanism to the SMB File Service.
- SMB File Service: Provides and maintains a secure and consistent file service.
- Object Store: Stores files and directories.

Stakeholders and Interests: There are no additional Stakeholders.

Preconditions: The client and server computers are configured in a domain. An SMB file share has been created on the file server. User has successfully opened and obtained a handle to a directory on the file share using Open File SMB and used it to request change notifications.

Minimal Guarantees: No action is taken that affects other files exposed by the file server as a result of this operation.

Success Guarantee: The User receives notification of a change in the state of that directory.

Trigger: The SMB File Service sends a change notification to the SMB File Client for the monitored directory.

Main Success Scenario:

1. SMB File Client receives change notification for the monitored directory with an indication of the type of change that occurred in the Object Store as described in [MS-CIFS] sections 2.2.7.4 and 3.2.5.40.3, [MS-SMB] section 2.2.4.8, and [MS-SMB2] section 2.2.36.
2. SMB File Client supplies the results of the operation to the Application.

Extensions:

1. A protocol error code is returned from the SMB File Service:

- Application displays an error message corresponding to the Error code returned and the use case aborts.

4 System Context

This section describes the relationship between this system and its environment.

4.1 System Environment

Network Infrastructure: This system requires access to network services which support<4>:

- TCP/IP (IPv4 or IPv6).
- UDP/IP (IPv4 or IPv6, for the NFS Access Protocols).
- Domain Naming System (DNS) name resolution.

Domain DFS Namespaces: If the system instance is to support domain namespaces, a DFS Service instance MUST be running on all domain controller computers, because File Clients accessing a domain namespace will assume that all domain controllers are running a DFS Service, for the purposes of retrieving configuration information such as Group Policy [MS-GPSO].

Object Store: Both SMB and NFS File Services must have access to a hierarchical object store for persistence of files and namespace. The Object Store would typically be built on file system(s) available in the host OS, but the Object Store may also need to include functionality in addition to that provided by the file system. A File Service may need to implement additional logic to convert between the semantics of a particular File Access Protocol and semantics of the available Object Store. Some semantics implemented in Windows file systems are directly visible when using the SMB Access Protocols, and such wire visible behaviors are documented in [FSBO].

Case Sensitivity: The Object Store must support case insensitive operation for the SMB File Service, and should support case sensitive operation for the NFS File Service.

Accounts: If computers hosting File Clients and File Services are not joined to a domain, then user accounts must be configured across computers by some external mechanism.

4.2 System Assumptions and Preconditions

The following assumptions and preconditions must be satisfied for the File Access Services System to operate successfully:

System Availability: The File Access Services System must be installed on all the computers involved.

Domain Configuration: In a domain configuration, File Client and File Service have access to directory services provided by the domain.

Authentication Services: Authentication services as described in [MS-AUTHSO] are available to all File Clients and File Services.

RPC: Components of File Client and File Service that use Remote Procedure Call interfaces MUST have prerequisites specified in [MS-RPCE] section 1.5 satisfied.

Network Configuration: In order for system components running on different computers to communicate, the network services and infrastructure must be functional and configured such that required protocols, ports, and so on are remotely accessible.

Domain Functionality: For system functionality requiring a domain (as defined in [MS-DISO]) and directory services (as defined in [MS-ADSO]), at least one domain controller must be configured and accessible. Some functionality, such as domain-based DFS namespaces, may require an **Active Directory-style domain** as noted in [MS-DFSC].

Domain Functional Level: In order for an NFS File Service to authenticate users represented by AUTH_SYS credentials, this system must have available a domain at a functional level which supports S4U extensions<5>.

Directory Schema: In order for an NFS File Service to implement user name to ID mapping based on [RFC2307], there must be an LDAP store available, such as described in [MS-ADSO], which is configured to include schema elements <uidNumber> and <gidNumber> from [RFC2307].

4.3 System Relationships

This section defines the File Access Services System as a black box, and examines the relationships of the black box to components outside of the black box, in terms of system dependencies and system influences.

4.3.1 Black Box Relationship Diagram

The File Access Services System is shown in the following figure as a "black box," showing a high-level abstraction of the File Access Services System's major components, and the external systems and components with which the File Access Services System interacts.

The abstract components of the File Access Services System are as follows:

- File Service is an abstraction of the software running on a file server that provides remote file access to one or more Users. It can also be managed by one or more Administrators. Internally, it contains an object store to persist changes made to files through the file access protocols. The object store is an implementation-dependent local file system.
- File Client and Admin Client are abstractions of low-level protocol state and operating software that runs in application and administrative nodes, respectively. These Clients are considered part of the File Access Services System. They are used by Application and Admin Tool software to effect communication with the remote components of the File Access Services System, but they do not hide the details of the protocols from the Application and Admin Tool, and they do not represent the operating system kernel, which typically abstracts the details of protocol interaction. In many respects, File Client and Admin Client are identical in function; they are separated here for convenience in discussing file access protocols and file services administration protocols separately.
- **Application** and **Admin Tool** represent programs with which a User and an Administrator typically interact, respectively. It should be understood that this is a somewhat arbitrary distinction, and that many programs have both Application and Admin Tool features. For example, a graphical shell is frequently used directly by a user to perform personal file management tasks. This type of program is classified as an Application, but it certainly has some aspects of an Admin Tool.

The dotted lines model dependency and influences relationships between the File Access Services System and external systems and components, which will be examined further in sections 4.3.2 and 4.3.3. The File Access Services System has few true dependencies because of the diversity of environments that it can be deployed within. Thus, most external relationships are influences because the system will operate with reduced functionality if the external service is not present.

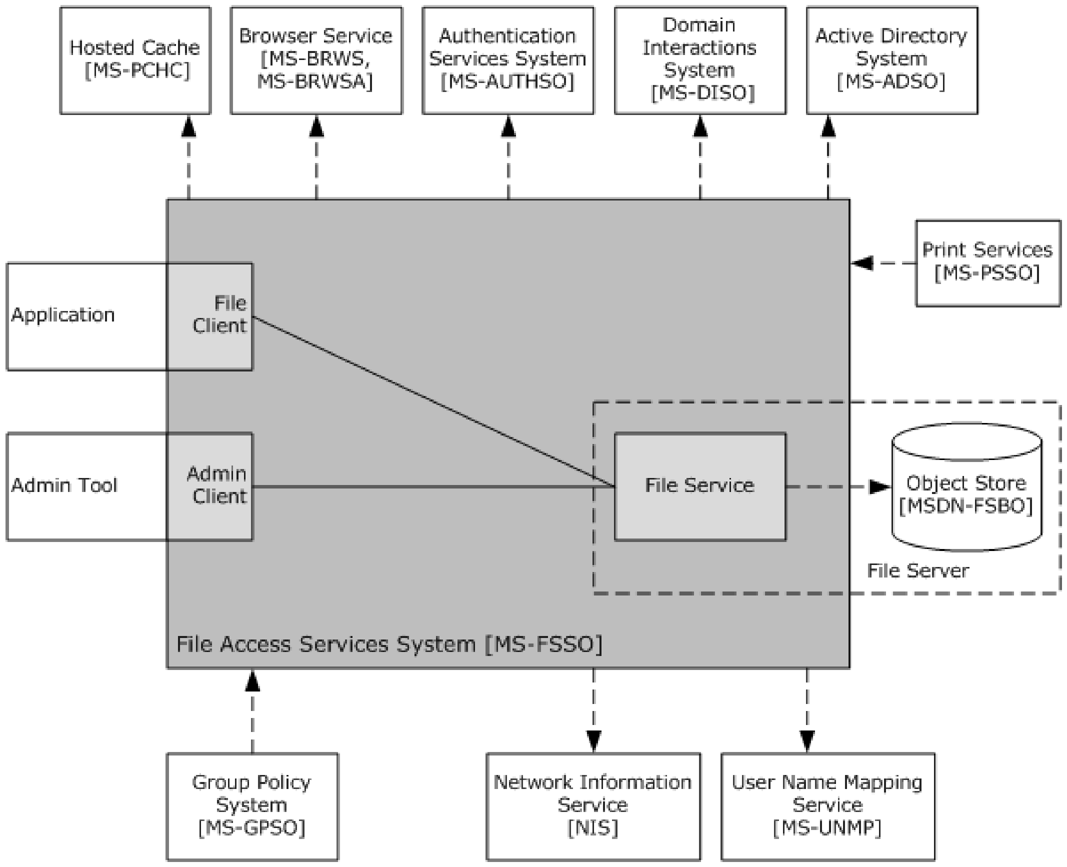

Hosted Cache [MS-PCHC]
Browser Service [MS-BRWS, MS-BRWSA]
Authentication Services System [MS-AUTHSO]
Domain Interactions System [MS-DISO]
Active Directory System [MS-ADSO]
Print Services [MS-PSSO]
Application
File Client
Admin Tool
Admin Client
File Service
Object Store [MSDN-FSBO]
File Server
File Access Services System [MS-FSSO]
Group Policy System [MS-GPSO]
Network Information Service [NIS]
User Name Mapping Service [MS-UNMP]

Figure 5: File Access Services System black box diagram

4.3.2 System Dependencies

The File Access Services System depends on the following external systems and components:

- Object Store: For storing files and metadata.

The File Access Services System also depends on certain network protocols. These are described in greater detail in section 5.1.

- The SMB Access Protocols can run over NetBEUI 3.0, NetBIOS over IPX, NetBIOS over TCP, or directly on TCP.
- [MS-RAP] does not operate over [MS-SMB2]; it depends upon either [MS-CIFS] or [MS-SMB].
- [MS-WKST], [MS-SRVS], and [MS-DFSNM] depend on [MS-RPCE] mapping to named pipes (that is, the SMB Access Protocols).
- [MS-FSRM] depends on [MS-DCOM] mapping to TCP/IP.
- The NFS Access Protocols require Sun RPC ([RFC5531]), using either the TCP or UDP mapping.

The following external systems depend on the File Access Services System:

- Print Services ([MS-PSSO]): For transferring a print image file to the print server.

- Group Policy ([MS-GPSO]): For downloading group policy information from the SYSVOL share to group policy client, which behaves for this purpose as a File Access Services Application.

4.3.3 System Influences

The File Access Services System can be influenced by the external systems and components shown in the following table.

External entity	File Access Services System depends on external entity for...	Consequences if absent
Browser Service	A list of computers in a workgroup or domain environment and their associated services.	Cannot discover services associated with other computers.
Authentication Services System	Authenticating client and server principals.	Centralized identity management does not work if the Domain Interactions System is not available.
Active Directory System	Storing DFS namespace metadata.	Cannot create domain controller based DFS namespace.
Hosted Cache	Protocols and services used to implement BranchCache caching of file server content.	Branch caching does not function.
Group Policy System	Configuration of individual protocol capabilities within the File Access Services System.	Cannot centrally configure some functionality of the system.
Network Information Service	Enumerating sets of NFS Client computers as defined in the Netgroups table to be given specific NFS share access.	Cannot centrally configure some functionality of the system.
User Name Mapping Service	Enumerating identity mapping information for user and group accounts to convert between AUTH_SYS style identity and Windows account names.	Cannot centrally configure some functionality of the system.

Specific system influences are as follows:

- Identity conversion from credentials supplied by NFS File Client to credentials acceptable to the local authentication system on the NFS File Service (for example, [MS-AUTHSO]) and to the Object Store is done through [MS-UNMP] or an [RFC2307] compliant LDAP store (such as [MS-ADSO]).
- nisNetgroup object as defined in an [RFC2307] compliant LDAP store or netgroups defined in Network Information Service [NIS] are consumed by the NFS File Service to enforce client fencing on NFS File Shares.
- Group Policy enables the File Client to interact with the BranchCache Hosted Cache Service, which is off by default. This enables an [MS-SMB2] client to publish and retrieve content for a file that is located on an [MS-SMB2] share.

4.4 System Applicability

The File Access Services System is central to file and printer sharing and Group Policy distribution in Windows operating environments. If any of these functionalities are desired, then File Access Services must be deployed.

The system is most applicable in local-area networks, where network round-trip delay is small. However, the system can be operated satisfactorily on wide-area networks, with some loss of performance. In the important case of a branch (satellite) office, a hosted cache and associated caching helps compensate for message delay in communicating with the SMB file server, which is typically located centrally.

File Access Services System protocols support signing for message integrity, but they do not themselves support message encryption for privacy. Consequently, the system should normally be deployed on private networks, where messages are secure from eavesdropping. The use of Virtual Private Network technology is recommended when File Access Services systems are deployed using the public Internet.

4.5 System Versioning and Capability Negotiation

The File Access Services System does not define any versioning or capability negotiation beyond what is described in the member protocol specifications, as listed in section 2.2. However, the evolution of protocols through various releases of Windows has resulted in some important changes in the File Access Services System. These are summarized as follows.

The current File Access Services System evolved from earlier systems for remote file access, including the Microsoft LAN Manager. These early systems did not have a general RPC transport available to them, and instead defined protocol specific methods for encoding what would later be understood to be remote function calls. The Remote Administration Protocol as defined in [MS-RAP] is such a protocol.

With the introduction of Windows NT 3.1 operating system, an RPC transport as defined in [MS-RPCE] became available to implementers of the File Access Services System. Rather than continuing to extend the Remote Administration Protocol, the new Server Service Remote Protocol, as specified in [MS-SRVS], was defined, replacing the use of the Remote Administration Protocol within the File Access Services System between clients and servers based on the new platforms. Support for the Remote Administration Protocol was maintained, however, for interoperability with pre-RPC platforms including Windows 95 operating system.

In the Windows 7 operating system platform, server support for the Remote Administration Protocol has largely been deprecated. Clients within File Access Services systems including Windows 7 servers must use the Server Service Remote Protocol for operations such as enumerating file shares. The exception to this is within the CIFS Browser Protocol as defined in [MS-BRWS], which continues to use the Remote Administration Protocol to retrieve the file and print servers on the local subnet. This does not have an equivalent in the Server Service Remote Protocol.

A similar evolution has occurred within the SMB Access Protocols. The Common Internet File System, specified in [MS-CIFS], extended with the Server Message Block (SMB) Version 1.0 Protocol, specified in [MS-SMB], has origins in the same early systems for remote file access as does the Remote Administration Protocol. The Server Message Block (SMB) Version 2.0 Protocol, specified in [MS-SMB2], preserves the functionality of the earlier SMB Protocols, while consolidating remote APIs and implementing a more efficient packet structure.

SMB Version 2.1 offers functionality relevant to the Peer Content Caching and Retrieval: Hosted Cache Protocol [MS-PCHC] introduced in the Windows 7 platform.

The File Access Services System has evolved in its methods for integrating File Services into unified namespaces. The Distributed File System protocols, the Distributed File System (DFS): Referral Protocol [MS-DFSC] and Distributed File System (DFS): Namespace Management Protocol [MS-DFSNM], were introduced in Windows NT 4.0 operating system Service Pack 2 (SP2). The **DFS Referral** Protocol is an extension to the SMB Access Protocols, and is required in order to traverse DFS link on share which host DFS namespaces. A File Access Services System client who does not implement the DFS Referral Protocol, such as Windows NT 3.1, will only be able to access content local to the namespace share.

In the first version of DFS, Standalone DFS namespace were supported. With Windows 2000 operating system a new domain-based DFS namespace type was introduced, with extensions to the DFS Referral Protocol that are required in order to locate and navigate domain-based DFS namespace.

4.6 System Vendor-Extensible Fields

The File Access Services System does not define any vendor-extensible field; neither at the system level nor at the individual protocol level.

5 System Architecture

This section describes the basic structure of the system and the interrelationships among its parts, consumers, and dependencies.

5.1 Abstract Data Model

This section describes the conceptual data organization the system maintains to provide its functionality. The data model described in this section can be implemented in a variety of ways. This specification does not prescribe any specific implementation technique.

The purpose of this section is to document synchronized common elements from [MS-CIFS] and [MS-SMB2] that are exchanged between the File Access Services System member protocols and/or external systems in order to support operations involving more than one protocol.

The sharing of state happens internal to the server implementation. Compatible implementations can use any mechanism of their choice to perform the sharing, as long as it satisfies the requirements of this section and the protocol specifications. For example, an implementation could use a single store that is shared across all the protocols or it could use multiple stores (one per protocol) and build a synchronization mechanism to maintain consistency across all the stores, as long as that synchronization mechanism ensures that all atomicity requirements are observed and that two protocols that share state always see an identical view of any common elements.

The abstract data model (ADM) for the File Access Services System is best described by separating the client portion of the system from the server.

Unless otherwise specified, elements are not persisted, and have no synchronization rules.

5.1.1 Client Data Model

The following figure shows the ADM for the client portion of the File Access Services System.

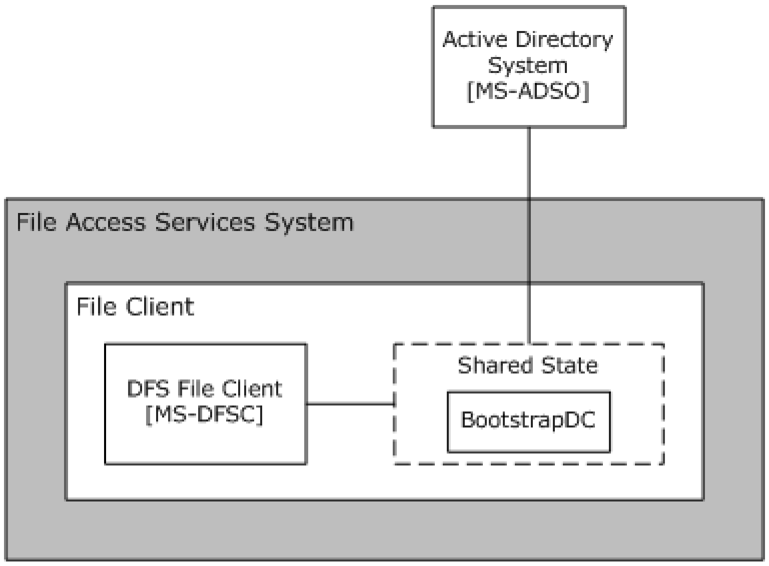

Active Directory System [MS-ADSO]
File Access Services System
File Client
DFS File Client [MS-DFSC]
Shared State
BootstrapDC

Figure 6: Common elements exchanged between file client and Active Directory

There are no common elements maintained between the following:

- NFS File Client and SMB File Client.
- NFS File Client and **DFS File Client**.
- SMB File Client and DFS File Client.

The following sections describe additional common-element relationships.

5.1.1.1 Common Elements Exchanged Between DFS File Client and Active Directory System

5.1.1.1.1 BootstrapDC

- Purpose: The purpose of this entity is to identify the domain controller from which to obtain a list of domain referrals and **DC host names** for a domain.
- Operations and Rules: This element value is retrieved as described in [MS-DISO] section and defined in [MS-DFSC] section 3.1.1.

5.1.2 Client and Server Data Model

The following figure shows the ADM for the common elements between the File Client and File Server of the File Access Services System.

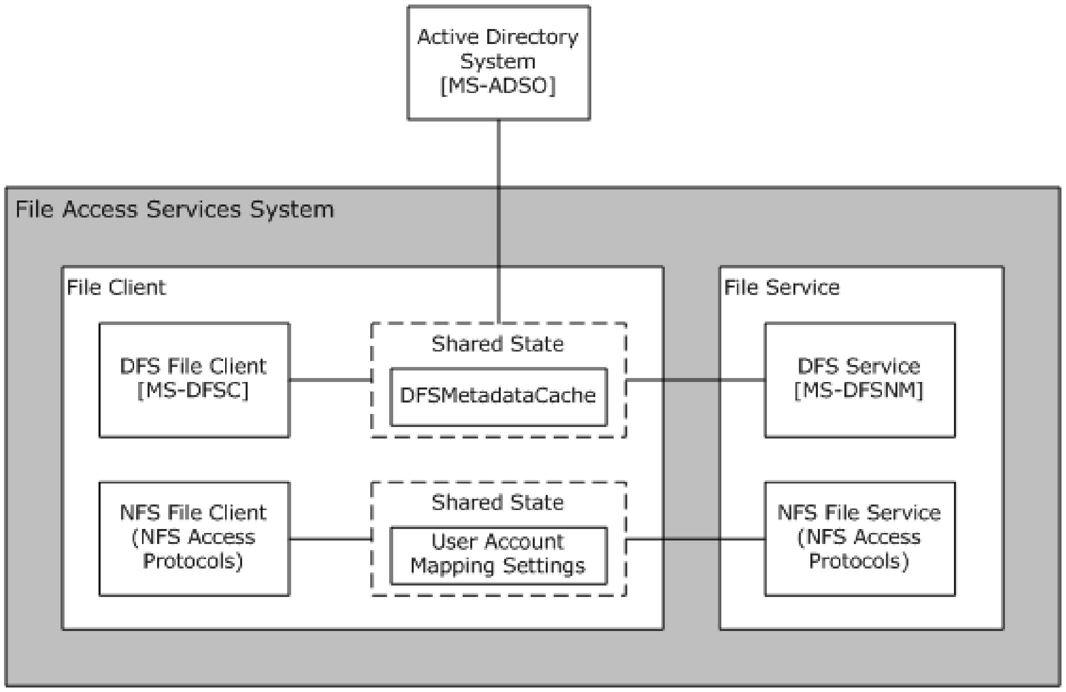

Active Directory System [MS-ADSO]
File Access Services System
File Client
DFS File Client [MS-DFSC]
Shared State
DFSMetadataCache
NFS File Client (NFS Access Protocols)
Shared State
User Account Mapping Settings
File Service
DFS Service [MS-DFSNM]
NFS File Service (NFS Access Protocols)

Figure 7: Common elements exchanged between file client, Active Directory, and File Service

5.1.2.1 Common Elements Exchanged Between NFS File Client and NFS File Service

The following section describes User Account Mapping settings exchanged between the NFS File Client and the NFS File Service.

5.1.2.1.1 User Account Mapping Settings

The following sections describe individual User Account Mapping settings that affect the behavior of the NFS File Client.

5.1.2.1.1.1 User Name Mapping Server Lookup Enabled

- Purpose: Specifies whether the NFS File Client should use a User Name Mapping Server [MS-UNMP] as a source of user account mappings.
- Operations and Rules: A Boolean value with "true" indicating that a User Name Mapping Server [MS-UNMP] should be used and "false" indicating that a User Name Mapping Server [MS-UNMP] is not to be used. The default is "false".

5.1.2.1.1.2 User Name Mapping Server List

- Purpose: Specifies the list of User Name Mapping Servers that the NFS File Client should use as a source of user account mappings.
- Operations and Rules: An array of string values, each of which specifies the server name or IP Address of a User Name Mapping Server [MS-UNMP]. This setting is used only when User Name Mapping Server Lookup Enabled is configured to "true". The default value is an empty string.

5.1.2.1.1.3 LDAP Lookup Enabled

- Purpose: Specifies whether the NFS File Client should use LDAP or Active Directory as a source of user account mappings.
- Operations and Rules: A Boolean value with "true" indicating that the LDAP server should be used and "false" indicating that LDAP is not to be used. The default is "true".

5.1.2.1.1.4 LDAP Server

- Purpose: Specifies the name of the LDAP server or Active Directory domain to use for user account mapping lookups by the NFS File Client.
- Operations and Rules: String value specifying the name or IP Address of the LDAP server to use for user account mapping lookups. This setting is used only when LDAP Lookup Enabled is configured to "true". The default is an empty string.

5.1.2.2 Common Elements Exchanged Between DFS File Client, DFS Service, and Active Directory System

The following section describes DFS metadata which is exchanged between the DFS File Client and the DFS Service via the Active Directory system.

5.1.2.2.1 DFSMetadataCache

- Purpose: This represents the DFS metadata of DFS Namespaces for which the server is a root target.
- Operations and Rules: Defined in [MS-DFSC] section 3.2.1 and [MS-DFSNM] section 3.1.1.

5.1.3 Server Data Model

The following figure shows the ADM for the server portion of the File Access Services System.

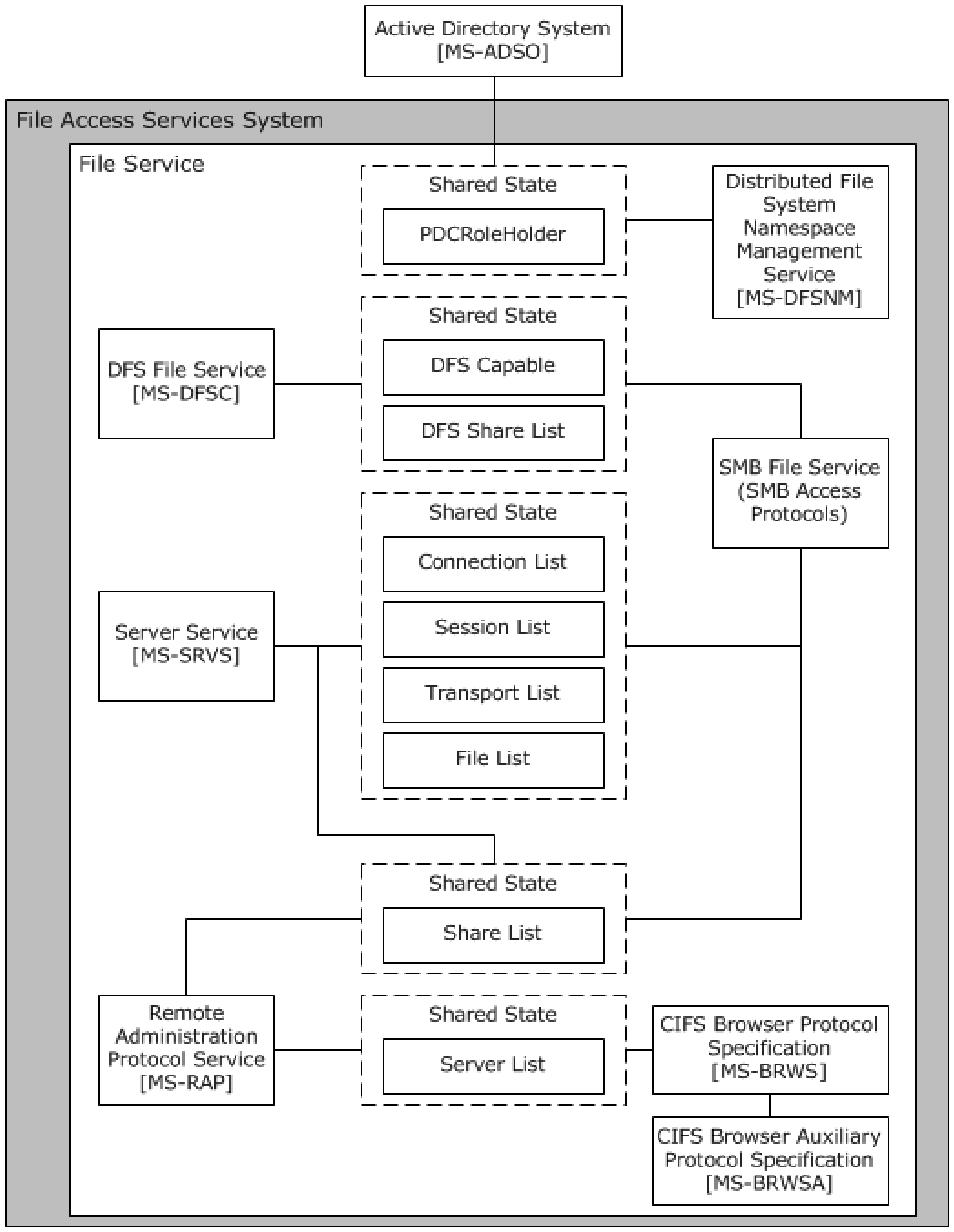

Active Directory System [MS-ADSO]
File Access Services System
File Service
Shared State
PDCRoleHolder
Distributed File System Namespace Management Service [MS-DFSNM]
Shared State
DFS Capable
DFS Share List
DFS File Service [MS-DFSC]
SMB File Service (SMB Access Protocols)
Shared State
Connection List
Session List
Transport List
File List
Server Service [MS-SRVS]
Shared State
Share List
Remote Administration Protocol Service [MS-RAP]
Shared State
Server List
CIFS Browser Protocol Specification [MS-BRWS]
CIFS Browser Auxiliary Protocol Specification [MS-BRWSA]

Figure 8: Common elements exchanged between File Service components of File Access Services System

5.1.3.1 Common Elements Exchanged Between DFS Service and Active Directory System

The following section describes common elements exchanged between the DFS Service and the Active Directory System.

5.1.3.1.1 PDCRoleHolder

- Purpose: Identify the primary domain controller for domain-based DFS namespaces.
- Operations and Rules: Defined in [MS-DFSNM] section 3.1.1. This value is retrieved through the algorithm described in [MS-DISO] section 5.4.5.2.

5.1.3.2 Common Elements Exchanged between DFS Service and SMB File Service (SMB Access Protocols)

The following sections describe common elements exchanged between the DFS Service and the SMB File Service.

5.1.3.2.1 DFS Capable

- Purpose: This data element represents whether or not the SMB File Service is currently DFS capable.
- Operations and Rules: Element value is updated by the DFS Service. See [MS-DFSC] section 3.2.3 for details and references.

5.1.3.2.2 DFS Share List

- Purpose: This element is a list of SMB share objects that are DFS aware.
- Operations and Rules: This data element is maintained by the DFS Service and consumed by the SMB File Service. See [MS-DFSC] section 3.2.3, [MS-CIFS] section 3.3.4.4, and [MS-SMB2] section 3.3.4.9.

5.1.3.3 Common Elements Exchanged between Server Service Remote Protocol and SMB File Service (SMB Access Protocols)

5.1.3.3.1 Connection List

This element is a list of connection objects obtained by invoking underlying server events, as specified in [MS-CIFS] section 3.3.4.15 and [MS-SMB2] section 3.3.4.19, each describing a single active tree connect from an SMB File Client to the SMB File Service. Connection objects are created and maintained by the SMB File Service, and consumed by the Server Service Remote Protocol [MS-SRVS] for the purpose of servicing **NetrConnectionEnum** requests (see [MS-SRVS] section 3.1.4.1). The server must maintain connection objects in order by always adding new connection objects to the end of the Connection List.

Connection objects are created when an SMB File Client establishes a tree connection to the SMB File Service, and discarded from the Connection List when the tree connection is removed. For more information on tree connects, see [MS-SMB2] sections 2.2.9 to 2.2.11, [MS-SMB] sections 2.2.4.7.1 and 2.2.4.7.2, and [MS-CIFS] sections 2.2.4.55 and 2.2.4.51.

5.1.3.3.1.1 Connection Object Elements

Connection object elements are maintained by the SMB File Service.

For more information on the elements, see [MS-SRVS] sections 2.2.4.1 to 2.2.4.5, [MS-SMB2] section 3.3.1.9, and [MS-CIFS] section 3.3.1.6. To identify the corresponding elements, see [MS-SMB2] section 3.3.4.19 and [MS-CIFS] section 3.3.4.15. For more information on the initialization of the elements, see [MS-SMB2] section 3.3.5.7 and [MS-CIFS] section 3.3.5.40.

Connection object element name	Description	Operations and rules
<Type>	Initialized to the type of the connection when the connection object is initially created.	Static once initialized.
<NumberOfOpens>	Specifies the number of files that are currently opened using the connection.	Updated by SMB File Service.
<NumberOfUsers>	Specifies the number of users that are currently sharing the connection.	Updated by SMB File Service.
<CreationTime>	The value is initialized to a time such that [MS-SRVS] can derive the number of seconds that have elapsed since the connection was established.	Static once initialized.
UserName	The name of the user who established the connection.	Static once initialized.

5.1.3.3.2 Session List

The **Session** List is a list of session List objects obtained by invoking the underlying server events, as specified in [MS-CIFS] section 3.3.4.14 and [MS-SMB2] section 3.3.4.18, each describing a single session. Initialization and update of session objects and all included elements are driven by the SMB File Service (SMB Access Protocols). Element data is consumed by [MS-SRVS] to service **NetrSessionEnum** requests; see [MS-SRVS] section 3.1.4.5.

Session objects are created when an SMB File Client establishes a session to an SMB File Service. The server must maintain all active sessions in the Session List and sort them in the order they are created. For session setup information see [MS-SMB2] section 2.2.5, [MS-SMB] section 2.2.4.6.1, and [MS-CIFS] section 2.2.4.53.

Session objects are deleted from the Session List when a client logs off (for example, SMB2 LOGOFF request, [MS-SMB2] section 2.2.7), or through **NetrSessionDel** [MS-SRVS] section 3.1.4.6.

5.1.3.3.2.1 Session Object Elements

For more information on session object elements see [MS-SRVS] sections 2.2.4.11 to 2.2.4.20, [MS-SMB2] section 3.3.1.8, and [MS-CIFS] section 3.3.1.5. To identify the corresponding elements, see [MS-SMB2] section 3.3.4.18 and [MS-CIFS] section 3.3.4.14. For more information about the initialization of the elements, see [MS-SMB2] section 3.3.5.5.1 and [MS-CIFS] section 3.3.5.43.

Session object element name	Description	Operations and rules
<ClientName>	Specifies the name of the Client computer that established the session.	Static once initialized.
<UserName>	The name of the user who established the session.	Static once initialized.
<NumberOfOpens>	Specifies the number of files that are currently opened using the session.	Incremented/decremented by SMB File Service when file handles are opened/closed by clients.
StartTime	Specifies the time of creation	Static once initialized.

Session object element name	Description	Operations and rules
	of the session object.	
<LastActiveTime>	Specifies the time the session object was last used; used to derive idle time.	Updated by SMB File Service when there is activity on the session.
UserFlags	Specifies how the user established the session. Values defined in [MS-SRVS] section 2.2.2.3.	Static once initialized.
<ClientLanmanTypeName>	Specifies the type of client that established the session. Values defined in [MS-SRVS] section 2.2.2.1.	Static once initialized.
<TransportName>	Specifies the name of the transport used to establish the session.	Static once initialized.

5.1.3.3.3 Transport List

A Transport List is a list of Transport objects obtained by invoking the underlying server events, as specified in [MS-CIFS] section 3.3.4.17 and [MS-SMB2] section 3.3.4.21, each representing a single Transport used for binding the server. Transport objects can be altered by [MS-SRVS] to determine which transports can be used by the SMB File Service (SMB Access Protocols) to accept incoming connection requests. The SMB File Service MUST be notified of the addition and removal of transport objects.

The server maintains all available transport objects in the Transport List as specified in [MS-SRVS] section 3.1.1.2.

Entries in the list can be added using **NetrServerTransportAdd** (see [MS-SRVS] section 3.1.4.22) or **NetrServerTransportAddEx** (see [MS-SRVS] section 3.1.4.23) and can be removed using **NetrServerTransportDel** (see [MS-SRVS] section 3.1.4.25) or **NetrServerTransportDelEx** (see [MS-SRVS] section 3.1.4.26).

5.1.3.3.3.1 Transport Object Elements

For more details about transport object elements, see [MS-SRVS] sections 2.2.4.93 through 2.2.4.96, [MS-SMB2] section 2.1, and [MS-CIFS] section 2.1. For more information on the initialization of the elements, see [MS-SRVS] sections 3.1.1.2 and 3.1.3.

Transport object element name	Description	Operations and rules
<NumberOfClients>	Identifies the number of client connections using the transport.	Cannot be set through [MS-SRVS]. Updated by SMB File Service.
<Name>	Implementation specific name of a device implementing the transport.	

Transport object element name	Description	Operations and rules
<TransportAddress>	Identifies the address being used by this server on the transport.	
<NetworkAddress>	Identifies the address being used by the underlying network adapter for the transport.	Cannot be set through [MS-SRVS].
<Domain>	Identifies the name of the domain to which the server must announce its presence.	
<Password>	Identifies the credentials required to add the transport.	

5.1.3.3.4 File List

A File List is a list of file objects obtained by invoking the underlying server events as specified in [MS-CIFS] section 3.3.4.16 and [MS-SMB2] section 3.3.4.20, each representing a file currently open through one or more SMB File Clients against the SMB File Server.

File objects are created when a file is first successfully opened by a client through successful completion of a create or open request. The server must maintain file objects in the File List and sort in the order that they are created, and are discarded from the File List when the last client open of a file is closed (either by a client, or through administrative mechanisms such as **NetrFileClose** (see [MS-SRVS] section 3.1.4.4).

File object elements are initialized and maintained by the SMB File Service (SMB File Protocols) and consumed by [MS-SRVS] in order to service **NetrFileEnum** (see [MS-SRVS] section 3.1.4.2 or **NetrFileGetInfo** [MS-SRVS] section 3.1.4.3) requests.

For information on create and open requests, see [MS-CIFS] section 2.2.4.64, [MS-SMB] sections 2.2.4.9.1 to 2.2.4.1.2, and [MS-SMB2] sections 2.2.13 and 3.3.5.9.

5.1.3.3.4.1 File Object Elements

For information about file object elements, see [MS-SRVS] sections 2.2.4.6 to 2.2.4.7, [MS-SMB2] section 3.3.1.10, and [MS-CIFS] section 3.3.1.7. To identify the corresponding elements, see [MS-SMB2] section 3.3.4.20 and [MS-CIFS] section 3.3.4.16. For more information about the initialization of the elements, see [MS-SMB2] section 3.3.5.9 and [MS-CIFS] section 3.3.5.5.

File object element name	Description	Operations and rules
<IdentificationNumber>	A session unique number assigned to the file when it was opened.	Static once initialized.
<PermissionMask>	Identifies the access permissions associated with the file open.	Static once initialized.
NumberOfLocks	Identifies the number of byte range locks currently granted for the open file.	Incremented/decremented by SMB File Service when lock or unlock requests are successfully processed.

File object element name	Description	Operations and rules
<Pathname>	Identifies the path of the opened file.	Static once initialized.
<UserName>	Identifies the name of the user whose credentials were used to open the file.	Static once initialized.

5.1.3.3.5 Server Properties

Additional information can be found in [MS-SRVS] sections 2.2.4.41 to 2.2.4.44.

In the following descriptions, sv10x_<fieldname> denotes a field available in one or more of the SERVER_INFO_10x structures where x can be 0, 1, 2, or 3. For further information on SERVER_INFO_100 see [MS-SRVS] section 2.2.4.38, for SERVER_INFO_101 see [MS-SRVS] section 2.2.4.39, for SERVER_INFO_102 see [MS-SRVS] section 2.2.4.40, and for SERVER_INFO_103 see [MS-SRVS] section 2.2.4.41.

In the following descriptions, svxxx_<fieldname> denotes a field available in one or more of the SERVER_INFO_xxx structures where the field may be set individually or as one of a group depending on the structure used. For further information see SERVER_INFO_599, [MS-SRVS] section 2.2.4.44.

Unless otherwise specified, element values can be queried using **NetrServerGetInfo**, [MS-SRVS] section 3.1.4.17, and set using **NetrServerSetInfo**, [MS-SRVS] section 3.1.4.18.

The element values here control certain behaviors of the SMB File Service, which must be made aware of changes to these values in an implementation specific manner. Values must be stored persistently in a way that allows them to remain in effect after the system is reinitialized, in an implementation specific manner.

5.1.3.3.5.1 Server Properties Object Elements

Server properties element name	Field mapping(s) and description	Operations and rules
<AcceptDownLevelApis>	**sv599_acceptdownlevelapis** Specifies whether the SMB File Service accepts method calls from previous-generation LAN Manager clients.	Cannot be set by [MS-SRVS].
AutoDisconnect	**sv10x_disc** Specifies whether the auto-disconnect feature is enabled and if so, the interval (in minutes) the session should be idle before the SMB File Service initiates a disconnect for the session.	See also [MS-SMB2] section 3.3.2.3. See section 6.4.1 of this document.
<Capabilities>	**sv103_capabilities** Specifies which of the capabilities of the SMB File Service are available and active.	Cannot be set by [MS-SRVS].
<EnableFcbOpens>	**sv599_enablefcbopens** Specifies whether several File Control Blocks (FCBs) are placed in a single location accessible to	

Server properties element name	Field mapping(s) and description	Operations and rules
	the server.	
<EnableForcedLogoff>	**sv599_enableforcedlogoff** Specifies whether the server must force a client to disconnect, even if the client has open files, after the client's logon time has expired.	
<EnableRawServer MessageBlocks>	**sv599_enableraw** Specifies whether the SMB File Service processes raw Server Message Blocks.	
<EnableOplocks>	**sv599_enableoplocks** Specifies whether the SMB File Service allows clients to use **opportunistic locks** on files.	
<EnableSharedNetDrives>	**sv599_enablesharednetdrives** Specifies whether the server allows redirected server drives to be shared.	
EnableSoftCompatibility	**sv599_enablesoftcompat** Specifies the SoftCompatibility capability of the SMB File Service.	
<InitialConnectionTableEntries>	**sv599_initconntable** Specifies the initial size of the SMB File Service connection table in terms of the number of entries in the table.	
<InitialFileTableEntries>	**sv599_initfiletable** Specifies the initial size of the SMB File Service file table in terms of the number of entries in the table.	
<InitialSearchTableEntries>	**sv599_initfiletable** Specifies the initial size of the SMB File Service search table in terms of the number of entries in the table.	
<InitialSessionTableEntries>	**sv599_initsesstable** Specifies the initial size of the SMB File Service session table in terms of the number of entries in the table.	
InitialWorkItems	**sv5xx_initworkitems** Specifies the initial number of receive buffers, or work items, that the SMB File Service has available to process incoming requests.	Cannot be set by [MS-SRVS].
<IrpStackSize>	**sv5xx_irpstacksize** Specifies the number of stack locations that the SMB File Service allocated in I/O request packets (IRPs).	Cannot be set by [MS-SRVS].
<MaximumFreeConnections>	**sv5xx_maxfreeconnections**	

Server properties element name	Field mapping(s) and description	Operations and rules
	Specifies the maximum number of free connection blocks that are maintained per **endpoint**.	
<MaximumKeepSearch>	**sv5xx_maxkeepsearch** Specifies the length of time, in seconds, that the SMB File Service retains information about incomplete directory search operations.	
<MaximumLinkDelay>	**sv5xx_maxlinkdelay** Specifies the maximum link delay, in seconds, for the SMB File Service.	
<MaximumMultiplexCount>	**sv5xx_maxmpxct** Specifies the maximum number of outstanding requests that any one client can send to the SMB File Service.	
<MaxNonPagedMemoryUse>	**sv5xx_maxnonpagedmemoryusage** Specifies the maximum amount of non-pageable memory, in bytes, that the SMB File Service can allocate at any one time.	
<MaxPagedMemoryUsage>	**sv5xx_maxpagedmemoryusage** Specifies the maximum amount of pageable memory, in bytes, that the SMB File Service can allocate at any one time.	
MaxTreeConnectsPerSession	**sv5xx_sessconns** Specifies the maximum number of tree connection that can be made to the SMB File Service using a single session.	
<MaxUsersPerSession>	**sv5xx_sessusers** Specifies the maximum number of users that can be logged on to the SMB File Service using a single session.	
<MaxWorkItems>	**sv5xx_maxworkitems** Specifies the maximum number of receive buffers, or work items, the SMB File Service can allocate before initiating flow control on the underlying transport.	
<MinimumFreeConnections>	**sv5xx_minfreeconnections** Specifies the minimum number of free connection blocks that are maintained per endpoint.	
<MinimumFreeWorkItems>	**sv5xx_minfreeworkitems** Specifies the minimum number of available receive work items that the SMB File Service requires to begin processing a server message block.	
<MinimumLinkThroughput	**sv5xx_minlinkthroughput**	

Server properties element name	Field mapping(s) and description	Operations and rules
ForOplocks>	Specifies the minimum link throughput, in bytes per second, for the SMB File Service to enable oplocks.	
MinimumReceiveQueueSize	**sv5xx_minrcvqueue** Specifies the minimum number of free receive work items the SMB File Service requires before it begins to allocate more.	
<OpenSearchLimit>	**sv599_opensearch** Specifies the number of search operations that can be carried out simultaneously.	
<OpensPerSession>	**sv599_sessopens** Specifies the number of files that can be open in one session.	
<OplockBreakResponseTime>	**sv5xx_oplockbreakresponsewait** Specifies the period of time, in seconds, that the SMB File Service waits for a client to respond to an **opportunistic lock break** request from the SMB File Service.	
<OplockBreakWaitTime>	**sv5xx_oplockbreakwait** Specifies the period of time, in seconds, to wait before timing out an opportunistic lock break request.	
<ScavengerTimeout>	**sv5xx_scavtimeout** Specifies the period of time in seconds that an implementation specific timer on the SMB File Service remains idle.	
<ServerRequestBufferSize>	**sv5xx_sizreqbuf** Specifies the size, in bytes, of each server buffer.	Cannot be set by [MS-SRVS].
<SessionsPerClient>	**sv5xx_sessvcs** Specifies the maximum number of sessions permitted per client.	Can only be set to 1.
<ThreadPriority>	**sv5xx_threadpriority** Specifies the priority of all SMB File Service threads in relation to the base priority of the SMB File Service process.	Cannot be set by [MS-SRVS].
<Users>	**sv10x_users** Specifies the upper bound of the number of users who can simultaneously establish a session to the SMB File Service.	
<WorkItemsForRawIO>	**sv5xx_rawworkitems** Specifies the number of special work items the SMB File Service uses for raw mode I/O.	

5.1.3.4 Common Elements Exchanged Between Server Service Remote Protocol, Remote Administration Protocol, SMB File Service (SMB Access Protocols), and DFS Service

The following sections describe common elements exchanged between the Server Service, Remote Administration, SMB File, and DFS services and protocols.

5.1.3.4.1 Share List

The share list is a list of share objects obtained by invoking the underlying server events, as specified in [MS-CIFS] section 3.3.4.12 or [MS-SMB] section 3.3.4.7, and [MS-SMB2] section 3.3.4.16, with each describing a single share.

Share objects are created in response to **NetrShareAdd** [MS-SRVS] section 3.1.4.7 messages, and can be manipulated using any of the NetrShareXxxx messages described in [MS-SRVS] section 3.1.4. SMB File Service must be notified, in an implementation specific manner, whenever share objects are created, modified, or deleted.

Each individual share object may be persistent, but this is not required by the system. The persistence of a share affects behavior within the MS-SRVS protocol as specified in [MS-SRVS] sections 3.1.1, 3.1.3, 3.1.4.9, and 3.1.4.13.

5.1.3.4.1.1 Share Object Elements

All elements are used by [MS-SRVS] and the SMB File Service, but different subsets of the elements are used by [MS-RAP] and [MS-DFSC].

For more details on share object elements, see [MS-SRVS] section 2.2.4.22 to 2.2.4.31, [MS-SMB2] section 3.3.1.6, [MS-CIFS] section 3.3.1.2, and [MS-RAP] section 2.5.6. To identify the corresponding elements, see [MS-SMB2] section 3.3.4.16 and [MS-CIFS] section 3.3.4.12. For more information about the initialization of the elements, see [MS-SMB2] section 3.3.4.13 and [MS-CIFS] section 3.3.4.9.

Share object element name	Description	Operations and rules
<Name>	Specifies a unique name for the resource described by the share object.	Not used by [MS-DFSC]. Static once initialized.
Type	Specifies whether the share object is used for access to a disk, a printer, a device, or for inter-process communication. For more details, see [MS-SRVS] section 2.2.2.4.	Not used by [MS-DFSC]. Static once initialized.
<Remark>	Identifies the number of byte range locks currently granted for the open file.	Not used by [MS-DFSC].
<MaxUses>	Specifies the maximum number of concurrent connections that the share object is permitted to accommodate.	Not used by [MS-RAP] or [MS-DFSC].
<CurrentUses>	Specifies the current number of connections to the share object.	Not used by [MS-RAP] or [MS-DFSC].

Share object element name	Description	Operations and rules
		Ignored by **NetrShareAdd** [MS-SRVS]. Updated by SMB File Service.
<Path>	Specifies the local object store path used to create the resource described by the share object.	Not used by [MS-RAP] or [MS-DFSC]. Static once initialized.
<SecurityDescriptor>	Specifies a **security descriptor** [MS-DTYP] section 2.4.6 used to determine access to the resource described by the share object.	Not used by [MS-RAP] or [MS-DFSC].
<Dfs>	Specifies whether the share object is configured for DFS [MS-DFSC].	Not used by [MS-RAP]. Initialized to false, updated by DFS Service [MS-DFSC].
<DfsRoot>	Specifies whether the share object is a root volume in the DFS tree structure.	Not used by [MS-RAP]. Initialized to false, updated by DFS Service [MS-DFSC].
<RestrictExclusiveOpens>	Specifies whether the share disallows exclusive file opens that deny reads to an open file.	Not used by [MS-RAP] or [MS-DFSC]. Initialized to false.
<EnableForcedShareDelete>	Specifies whether shared files in the specified share can be forcibly deleted.	Not used by [MS-RAP] or [MS-DFSC]. Initialized to false.
<EnableAccessBasedEnumeration>	Specifies whether the server filters returned directory entries based on the access permissions of the requesting client.	Not used by [MS-RAP] or [MS-DFSC]. Initialized to false.
<ForceLevel2Oplock>	Specifies whether the server should prevent issuing exclusive caching rights on this share.	Not used by [MS-RAP] or [MS-DFSC]. Initialized to false.
<EnableHashGeneration>	Specifies whether hash generation should be supported on this share for **branch cache** retrieval of data.	Not used by [MS-RAP] or [MS-DFSC]. Initialized to false.

5.1.3.5 Common Elements Exchanged Between [MS-RAP] and [MS-BRWS]

5.1.3.5.1 Server List

The Server List is a list of server objects with each describing a computer that exists on the network.

The Server List is initialized to an empty list and is updated as described in [MS-BRWS] section 3.3. The list can be queried as described in [MS-RAP] section 2.5.5.2 and 2.5.5.3 or [MS-BRWS] section 3.3.5.6.

5.1.3.5.1.1 Server Object Elements

Unless otherwise specified, elements are initialized as described in [MS-BRWS] section 3.3, and the format of the element is as described in [MS-RAP] section 2.5.5.4.2.

Server object element name	Description	Operations and rules
<Name>	Specifies the name of the server.	
<VersionHigh>	Specifies the most significant byte of the version of the operating system that is being run on the server.	The value is informational only and has no protocol significance.
<VersionLow>	Specifies the least significant byte of the version of the operating system that is being run on the server.	The value is informational only and has no protocol significance.
<Type>	Specifies the type of the server.	Values described in [MS-RAP] 2.5.5.2.1.
<Comment>	Specifies a comment for the server.	The value is informational only and has no protocol significance.

5.2 White Box Relationships

The following figures show the protocol layering relationships for the File Access Services System member protocols. The default relationship, indicated by a solid arrow is «is transported by». The «includes» stereotype means that a protocol includes a second protocol by reference (for example, [MS-SMB] includes [MS-FSCC]). The «extends» relationship means that a protocol is an extension of a base protocol. [MS-SMB] is based on, and extends [MS-CIFS]. Member protocols are shown in shaded boxes.

The following figure shows protocol relationships for SMB Access Protocols and related protocols. [MS-DFSC] comes into play for path name resolution during file open (Open File SMB use case). [MS-FSCC] provides details of data structures that are contained in SMB and SMB2 protocol headers.

[MS-RAP] is an administrative protocol that is transported by [MS-CIFS] and [MS-SMB], but not by [MS-SMB2]. In current Windows versions,<6> [MS-RAP] is used by the File Access Services System only to transport the NetServerEnum2 and NetServerEnum3 requests, which are used for obtaining a list of servers from the Browser service.

[MS-CIFS] and [MS-SMB] can be transported by NetBEUI, NetBIOS over IPX, and NetBIOS over TCP. [MS-SMB] can also be transported over TCP. The newer [MS-SMB2] is transported only by TCP, with or without a NetBIOS layer. Please see [MS-SMB2] section 2.1 Transport, [MS-SMB] section 2.1 Transport, and [MS-CIFS] section 3.2.4.2 for information on transport determination.

[MS-RAP] is transported only by [MS-CIFS] and [MS-SMB], and not by [MS-SMB2].

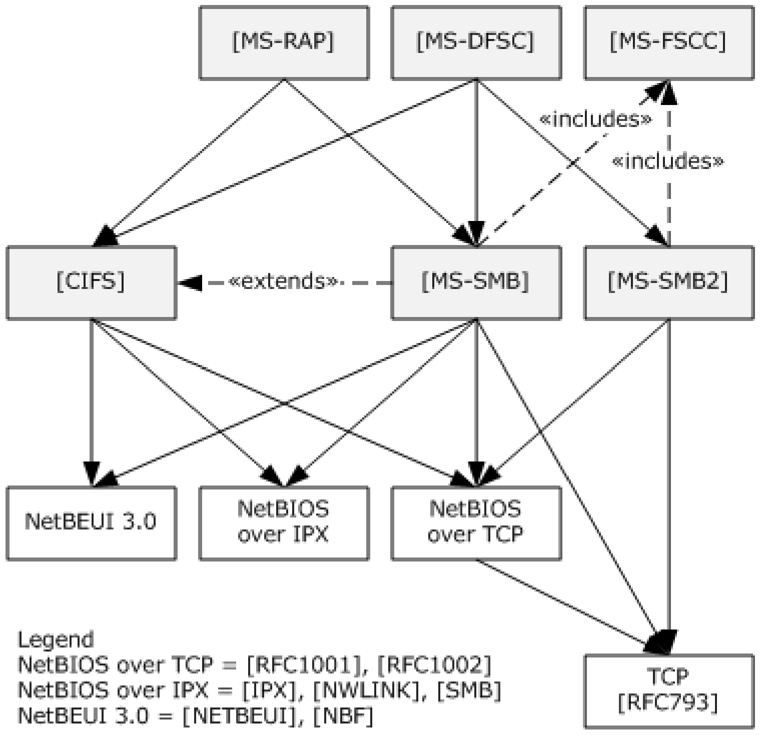
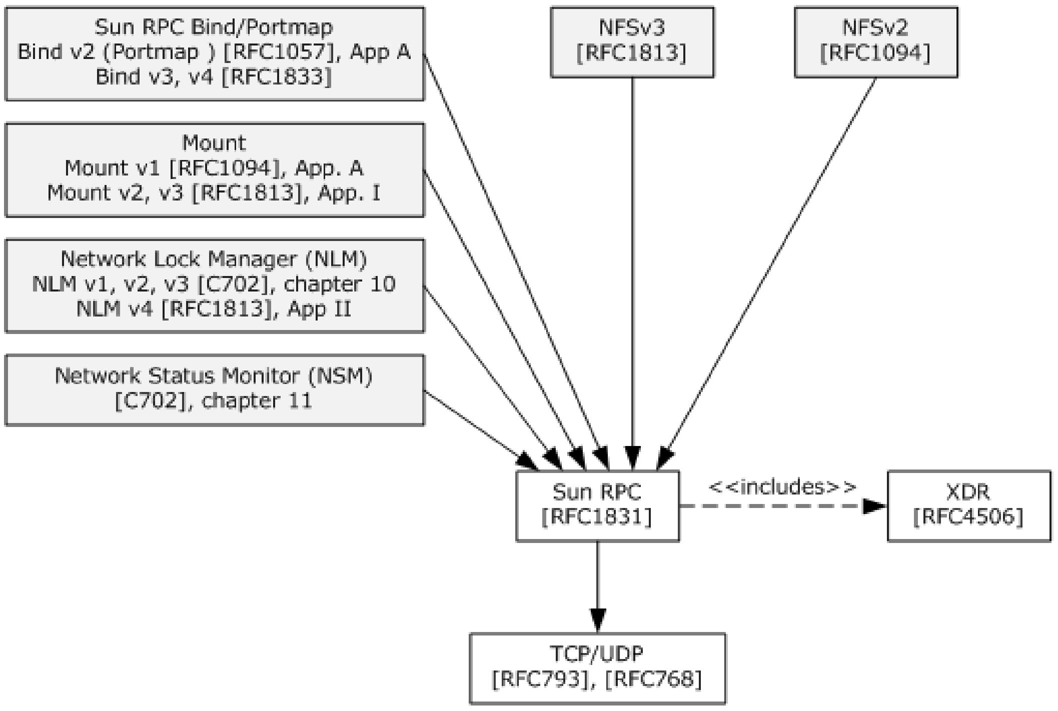


Figure 9: Protocol relationships for SMB and related protocols

The following figure shows the protocol relationships for NFS and related protocols. These protocols require TCP or UDP transport, but today they are used primarily with TCP.<7>



Notes
All protocols in the File Access Services System support both the TCP and UDP mappings of Sun RPC.
RFC4506 supersedes RFC1832 which supersedes RFC1014.
RFC1831 is an IETF standard, and is thus referenced for Sun RPC, instead of RFC1057, which is IETF Informational.

Figure 10: Protocol layering for NFS and related protocols

The following figure shows the protocol layering for file access related protocols that use RPC. [MS-WKST], [MS-SRVS], and [MS-DFSNM] use [MS-RPCE] over named pipes (protocol sequence string ncacn_np). Thus, the protocols can be transported over SMB Access Protocols, and cannot use any of the other RPC transports. [MS-FSRM] uses [MS-RPCE] over TCP only (protocol sequence string ncacn_ip_tcp).

Legend
Arrow labels represent [MS-RPCE] section 2.1 protocol sequence strings. They define the specific RPC transports that the protocol can operate over.

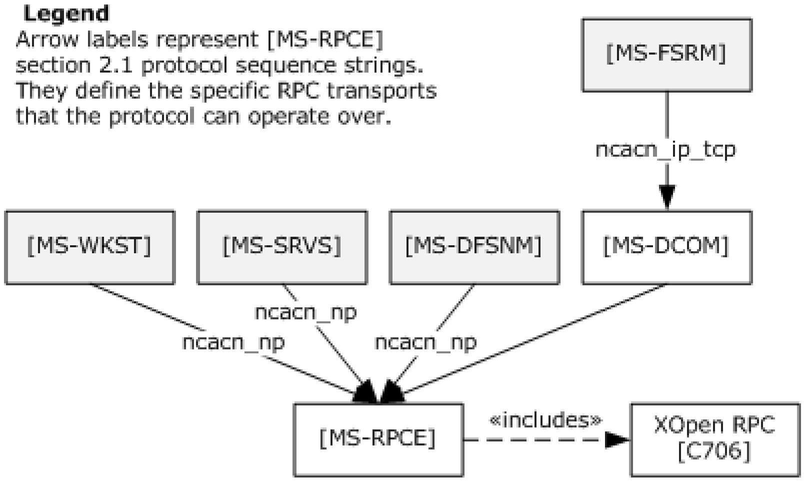


Figure 11: Protocol layering for RPC related protocols

5.3 Member Protocol Functional Relationships

This section provides the details on how each member protocol is utilized by the system. This includes the roles, interrelationships, and dependencies between the protocols.

5.3.1 Member Protocol Roles

This section describes all member protocol roles.

The **Remote Administration Protocol**, specified in [MS-RAP], is an administrative protocol whose function has largely been replaced by newer protocols. In the File Access Services System, Remote Administration Protocol is used as a discovery protocol. All Windows clients can discover servers by using Remote Administration Protocol to retrieve a list of servers from the Browser Service [MS-BRWS]. Remote Administration Protocol also supports certain client and server administration methods, such as SMB file share enumeration, but this functionality has been superseded by Workstation Service Remote Protocol, specified in [MS-WKST], and Server Service Remote Protocol, specified in [MS-SRVS].<8>

Although [MS-RAP] could potentially have been used by the File Access Services System for the administration of File Clients, it was never used for this purpose.

The Distributed File System (DFS) Namespace Referral Protocol, specified in [MS-DFSC], allows SMB File Clients to map paths in a virtual distributed namespace to paths on specific file servers.

The Common Internet File System Protocol ([MS-CIFS]), Server Message Block (SMB) Protocol ([MS-SMB]), and Server Message Block (SMB) 2.0 Protocol [MS-SMB2] are network file access protocols that support file sharing (remote access to an Object Store) between computers. These protocols have evolved considerably over the past decade or more. The original version of the Windows SMB protocol was first published within the IETF in 1997 and was called the Common Internet File System protocol. It included several sub-versions referred to as dialects that evolved over the years. SMB continued to evolve with each subsequent release of Windows, and is documented in [MS-SMB].<9>

Microsoft MSDN at times refers to these three protocols informally as "SMB", while the world at large often refers to them as "CIFS".

[RFC1094] and [RFC1813] support the sharing of file resources between computers.

Network Lock Manager (NLM) and Network Status Monitor (NSM) protocols [C702] are used in conjunction with the NFS protocols to provide support for file locking and service status monitoring.

The File System Control Codes, specified in [MS-FSCC], defines the network format of Windows data structures that are embedded in the SMB and SMB2 protocols, but not in the CIFS protocol.

The Distributed File System (DFS) Namespace Management Protocol, specified in [MS-DFSNM], is an administrative protocol that allows an Admin Client to configure and manage the virtual namespaces hosted by a DFS Service on a domain controller or file server.

The Workstation Service Remote Protocol, specified in [MS-WKST], is an administrative protocol that can be used to query and configure certain properties of the SMB File Client, such as performance information and limits. The File Access Services System as defined by Windows behavior does not actually use [MS-WKST] for any purpose, however Windows includes interfaces that allow third parties to use the protocol. Therefore, [MS-WKST] is included as a member protocol, but it does not appear in any of the system use cases or scenarios.

The Server Service Remote Protocol, specified in [MS-SRVS], is an administrative protocol that is used to query and configure certain properties of the SMB File Service on a server, such as active connections, sessions, shares, open files, and transport protocols.

The File Server Resource Manager Protocol, specified in [MS-FSRM], is an administrative protocol for managing file system (Object Store) policies (for example, directory quotas, file screens, classification properties, and classification rules) and **file management jobs** on a file server. File Server Resource Manager Protocol provides a set of Distributed Component Object Model (DCOM) interfaces for these tasks. It is included in the File Access Services System because many of these object store behaviors are visible through the file access protocols.

5.3.2 Member Protocol Groups

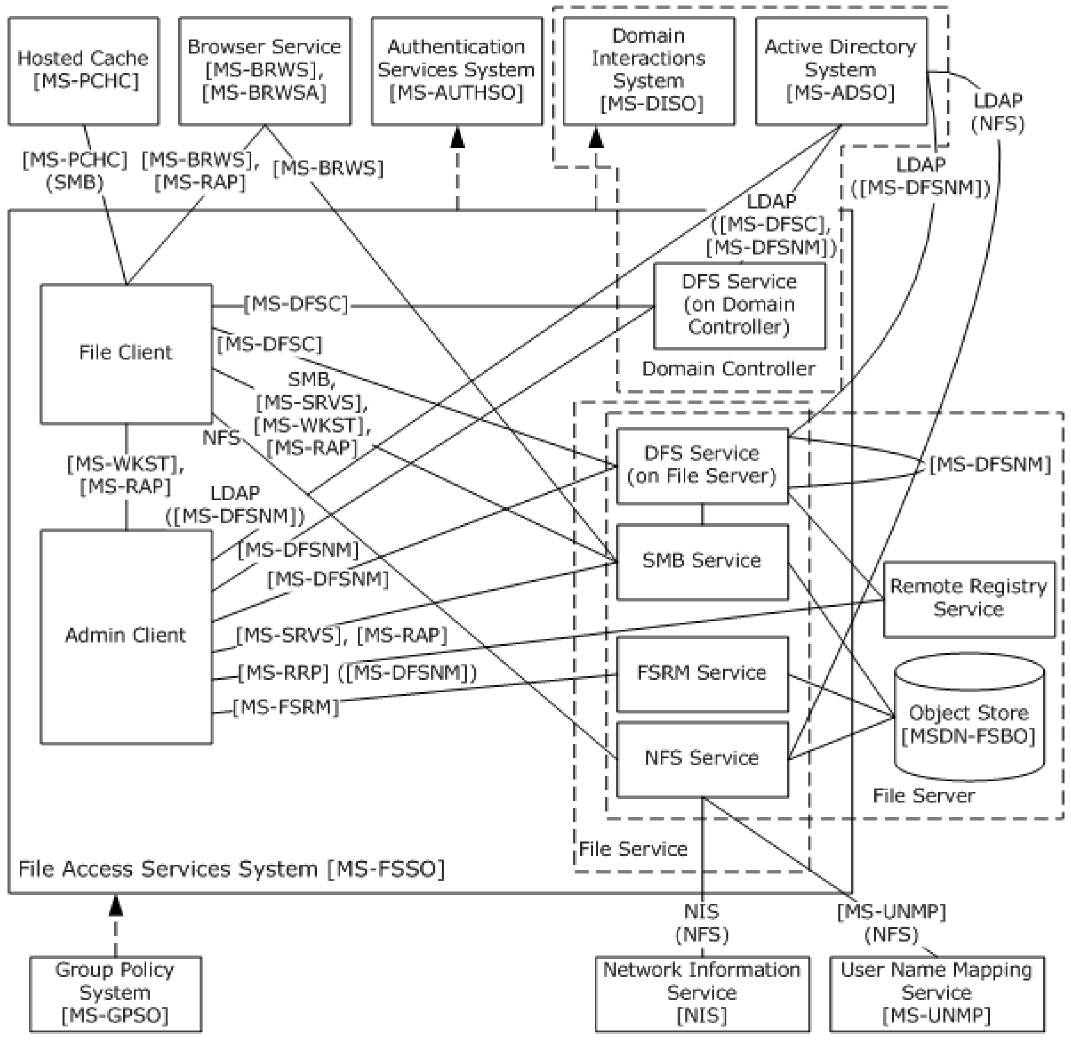
This section provides additional information regarding the roles of each of the protocol groups. Member protocol groups are as follows:

SMB Access Protocols, as specified in [MS-CIFS], [MS-SMB], [MS-SMB2], and [MS-FSCC]: The first three are successive generations of the SMB file access protocols, as described in section 5.3.1. [MS-FSCC] defines Windows data structures that are contained in [MS-SMB] and [MS-SMB2]. Collectively, these protocols are the primary solution for file sharing in Windows environments. Most SMB File Services support all of these protocols for maximum client compatibility.

NFS Access Protocols: NFS version 2 [RFC1094], NFS version 3 [RFC1813], and NLM/NSM [C702] collectively provide the primary solution for file sharing in heterogeneous computing environments. Most NFS File Services support all of these protocols for maximum client compatibility.

Administrative Protocols, as specified in [MS-WKST], [MS-SRVS], [MS-DFSNM], and [MS-FSRM].

5.4 System Internal Architecture



Hosted Cache [MS-PCHC]
Browser Service [MS-BRWS], [MS-BRWSA]
Authentication Services System [MS-AUTHSO]
Domain Interactions System [MS-DISO]
Active Directory System [MS-ADSO]
LDAP (NFS)
[MS-PCHC] (SMB)
[MS-BRWS], [MS-RAP]
[MS-BRWS]
LDAP ([MS-DFSNM])
LDAP ([MS-DFSC], [MS-DFSNM])
DFS Service (on Domain Controller)
Domain Controller
File Client
[MS-DFSC]
[MS-DFSC]
SMB, [MS-SRVS], [MS-WKST], [MS-RAP]
NFS
[MS-WKST], [MS-RAP]
LDAP ([MS-DFSNM])
DFS Service (on File Server)
[MS-DFSNM]
SMB Service
Remote Registry Service
Admin Client
[MS-DFSNM]
[MS-DFSNM]
[MS-SRVS], [MS-RAP]
[MS-RRP] ([MS-DFSNM])
[MS-FSRM]
FSRM Service
Object Store [MSDN-FSBO]
NFS Service
File Server
File Service
File Access Services System [MS-FSSO]
NIS (NFS)
[MS-UNMP] (NFS)
Group Policy System [MS-GPSO]
Network Information Service [NIS]
User Name Mapping Service [MS-UNMP]

Legend

- Solid lines represent the protocols used to communicate between components. They are labeled with the protocol name. Parentheses represent the Technical Document that causes the protocol to be exercised.
- Dotted lines represent dependencies on external systems.
- NFS and SMB represent the NFS Access Protocols and SMB Access Protocols, respectively.

Figure 12: File Access Services System internal architecture (white box)

The following table provides additional details about the nature of the communication between the components shown in the previous figure. The way to read this table is, "Component 1 uses Protocol to talk to Component 2 in order to accomplish Task".

Comp 1	Protocol	Comp 2	Task
Admin Client	LDAP [MS-DFSNM]	Active Directory System	The Administrative Client manages the initial creation, **access control lists**, and final deletion for the Distributed File System metadata stored in the Active Directory

Comp 1	Protocol	Comp 2	Task
			System. Through the access control lists, management of the metadata is delegated to DFS hosts running on file servers. Applicability: Domain
Admin Client	DFSNM	DFS Service (on file server)	Management of the Distributed File System: create/delete namespaces, manage links within namespaces, get/set properties and enumerate namespaces hosted by the file server. Applicability: Domain
DFS Service (on file server)	LDAP	Active Directory System	Update DFS metadata stored in Active Directory System, for example, add a link; also to pull metadata updates from the PDC when prompted through DFSNM
DFS Service (on file server)	DFSNM	DFS Service (on file server)	When performing an update to a namespace, the Distributed File System Service notifies peer replicas of the namespace, using [MS-DFSNM] NetrDfsSetInfo 3.1.4.1.5 (Level 101), that the update has occurred.
Admin Client	DFSNM	DFS Service (on domain controller)	The domain controller role of the Distributed File System provides a method for removing file server as replicas of a given domain namespace. This is commonly used when the file server is not available to perform the operation itself, as would be the case if the file server has catastrophically failed.
DFS Service (on Domain Controller)	LDAP (local loopback)	Active Directory System	Update DFS metadata stored in Active Directory System.
SMB Service	BRWS	Browser Service	Advertise the services. Applicability: Workgroup, domain
File Client	DISO	Domain Interaction Services	Locate a domain controller ([MS-DISO]) for the purpose of issuing trusted domain and domain referral requests, as specified in [MS-DFSC] 3.2.1.
File Client	BRWS	Browser Service	Locate backup browsers.
File Client	RAP	Browser Service	Enumerate file servers on the network by issuing NetServerEnum2 or NetServerEnum3 to a backup browser determined through [MS-BRWS].
File Client	SRVS	SMB Service	Retrieve server information, enumerate shares on a server, retrieve share information.
File Client	WKST	SMB Service	Applications may use WKST to discover the OS version of the remote computer, confirm the name of the host, and discover properties of the File Client present on the computer.
File Client	DFSC	DFS Service (on domain controller)	Domain Referral Request, DC Referral Request Applicability: Domain<10><11>

Comp 1	Protocol	Comp 2	Task
File Client	DFSC	DFS Service (on file server)	Root or Link Referral Request Applicability: Domain, workgroup, all Windows versions
File Client	CIFS, SMB, SMB2	SMB Service	Notification of a DFS link via an SMB Access Protocols Create operation failing with STATUS_PATH_NOT_COVERED.
File Client	CIFS, SMB, SMB2, NFS	SMB Service	Create (Open), Read, Write, Delete, Close, Query/Set attributes, Enumerate Directory, and other SMB and NFS Protocol Family operations.
Admin Client	WKST	File Client	Applications may use [MS-WKST] to discover the OS version of the remote computer, confirm the name of the host, and discover properties of the File Client present on the computer.
Admin Client	SRVS	SMB Service	Management of the SMB File Service: enumerate file shares, get/set share properties, create/delete shares and manage sessions and file handles established for users of the file service.
Admin Client	FSRM	SMB Service	Configuration of Quotas and File Screens on the file server Object Store.
NFS Service	UNMP	UNMP Server	Use of the User Name Mapping Protocol for Identity Mapping.
NFS Service	LDAP (RFC 2307)	Active Directory System	Use of an LDAP server using the RFC2307 schema for Identity Mapping as described in section 4.3.3.
NFS Service	SUNRPC	NIS	Use of the Network Information Service to discover netgroups as described in section 4.3.3.
Admin Client	RRP ([MS-DFSNM])	SMB Service	Use of the Remote Registry Protocol to control **ACL**s for delegation of administrative privilege in [MS-DFSNM].

5.5 Failure Scenarios

This section describes the common failure scenarios and specifies the system behavior in such conditions.

5.5.1 Connection Disconnected

A common failure scenario is an unexpected connection breakdown between the system and external entities. A disconnection can be caused by the network not being available, or by one of the communicating participants becoming unavailable. In the case where the network is not available, both participants remain active and expect the other party to continue the communication pattern specified by the protocol being executed at the time of the failure. Similarly, in the case where one of the participants is not available, the active participant expects the communication to proceed as specified by the protocol being executed.

Generally, a protocol detects a connection breakdown failure through either of the following methods:

- By using a timer object that generates an event if the corresponding participant has not responded within a reasonable time span.
- By being notified by the underlying protocol that the connection is disconnected.

When a connection disconnected event is detected, it causes the protocol to tear down all related communications and update any necessary data structures to maintain the system state.

Details about how each protocol detects a connection disconnected event, and how it behaves under this scenario, are provided in the specifications of the member protocols, as listed in section 2.2.

5.5.2 Internal Failures

The File Access Service System is not defensive against internal failures of its state, other than that described in the specifications of the member protocols. The components comprising the system mutually assume that each is authoritative at all times.

5.5.3 System Configuration Corruption or Unavailability

The system relies on the availability and consistency of its configuration data. Configuration consists of the data that determines the behavior of the system under specific conditions or for specific functionality. For example, the configuration can be used to enable or disable certain protocols, or whether the system can span across a network of computers.

If the configuration data is not available, the protocol that requires the configuration data may assume a default value.

6 System Details

This section contains the details that complete the descriptions in earlier sections of the document. These details are needed to understand and implement this system.

6.1 Architectural Details

This section contains a set of examples illustrating common uses of the File Access Services System. These examples provide more details of the system summary use cases introduced in section 3.3.4.1. The examples are as follows:

- Configure a File Service
- Find a file in a Workgroup
- Find a file in a Domain
- Two applications communicate using a shared file

Additionally, the examples include a number of referenced sub-scenarios, which are documented separately, in section 6.1.5. The sequence diagram examples in this section are shown in terms of messages defined in version 2 of SMB file access protocols [MS-SMB2] and version 3 of NFS file access protocols [RFC1813]. The semantics defined for these versions of the protocol can be mapped to other versions of the respective protocols.

6.1.1 Configure a File Service

This example illustrates the Configure File Service summary use case and each of the included detail use cases:

- Create Share SMB
- Create DFS Namespace
- Create DFS Link
- Create FSRM Quota
- Create FSRM File Screen

Configure File Service is illustrative of typical Admin interaction with the File Access Services System as the Admin provisions remotely accessible storage on a given file server.

6.1.1.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- The participating client and server computers must be configured to belong to the same Active Directory domain or workgroup.
- The specific path which will be provisioned for remote access must exist on the local object store of the file server.

6.1.1.2 Sequence of Events

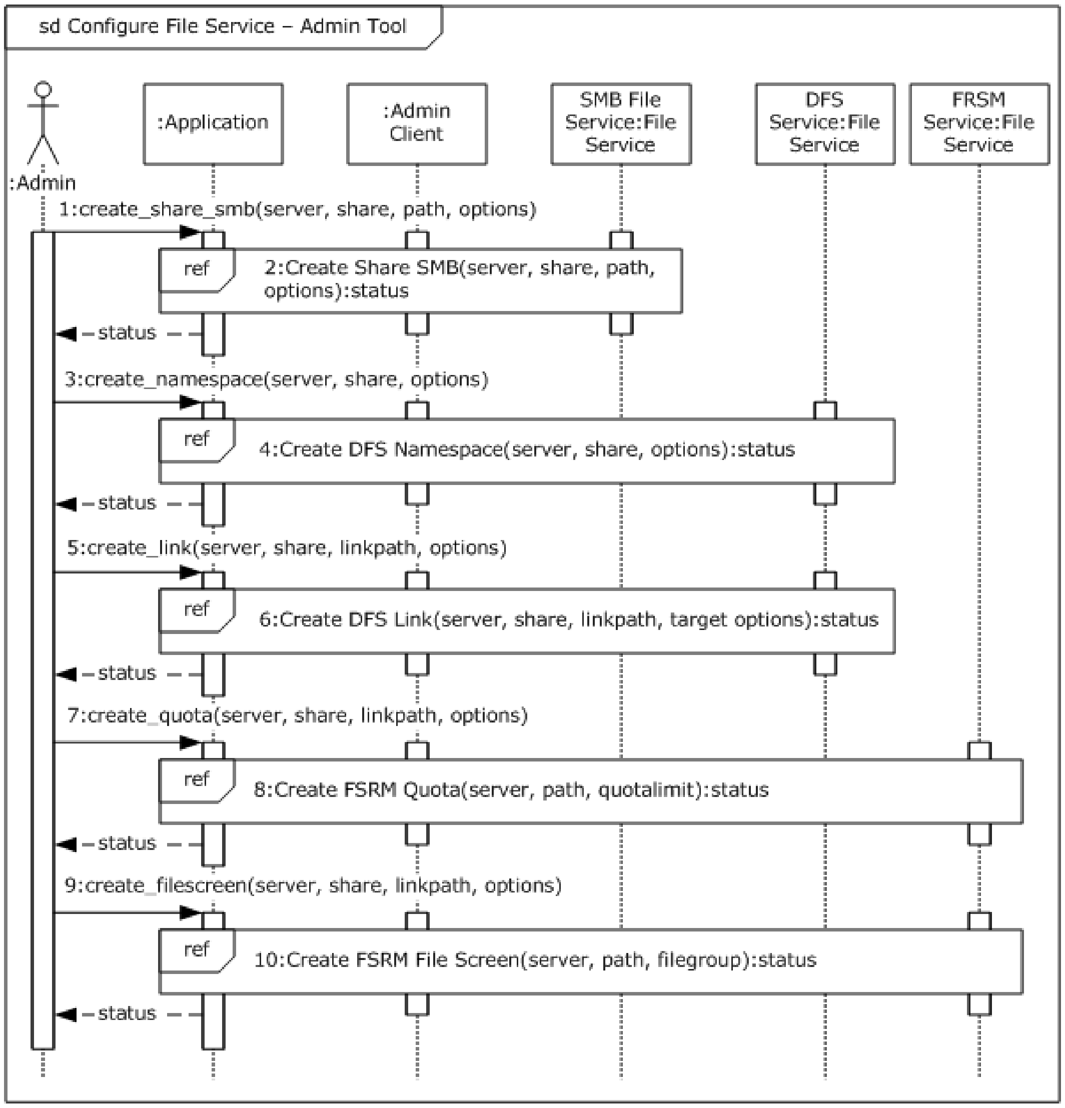

sd Configure File Service – Admin Tool
:Admin
:Application
:Admin Client
SMB File Service:File Service
DFS Service:File Service
FRSM Service:File Service
1:create_share_smb(server, share, path, options)
ref
2:Create Share SMB(server, share, path, options):status
status
3:create_namespace(server, share, options)
ref
4:Create DFS Namespace(server, share, options):status
status
5:create_link(server, share, linkpath, options)
ref
6:Create DFS Link(server, share, linkpath, target options):status
status
7:create_quota(server, share, linkpath, options)
ref
8:Create FSRM Quota(server, path, quotalimit):status
status
9:create_filescreen(server, share, linkpath, options)
ref
10:Create FSRM File Screen(server, path, filegroup):status
status

Figure 13: Sequence diagram for Configure File Service use case

1. The Admin requests the Application to create an SMB share, specifying the server, share name, local path on the object store, and various other options specific to the configuration of the share.

2. The Application makes use of the "Create Share SMB" sequence diagram, using the Admin supplied parameters, creating the share.

3. The Admin requests the Application to create a DFS namespace on the previously configured SMB share, specifying the server, share name, and various other options specific to the creation of the namespace.

4. The Application makes use of the "Create DFS Namespace" sequence diagram, using the Admin supplied parameters, creating the DFS namespace.

5. The Admin requests the Application to create a DFS link within the DFS namespace, specifying the server and share name of the namespace, the path at which the link should be created, target of the link, and various other options specific to creating the link.
6. The Application makes use of the "Create DFS Link" sequence diagram, using the Admin supplied parameters, creating the DFS link.
7. The Admin requests the Application to create an FSRM quota on the shared storage, specifying the server, local path on the object store, and desired quota limit.
8. The Application makes use of the "Create FSRM Quota" sequence diagram, using the Admin supplied parameters, instantiating the quota limit.
9. The Admin requests the Application to create an FSRM file screen policy on the shared storage, specifying the server, local path on the object store, and desired file group block.
10. The Application makes use of the "Create FSRM File Screen" sequence diagram, using the Admin supplied parameters, instantiating the file screen policy.

6.1.1.3 Final System State

The specified local path on the file server is available for SMB file access, is functioning as a DFS namespace with a single link, has a defined quota limit, and file screen policy is applied.

6.1.2 Find a File in a Workgroup

This example illustrates the Find File in Workgroup summary use case and each of the included detail use cases:

- Workgroup Query Backup Browser List
- Workgroup Query Computers List
- Bind Calling Context SMB Service
- Open File SMB
- Perform File Operation SMB Query FS Information
- Perform File Operation SMB Enumerate Directory
- Perform File Operation SMB Close
- Perform File Operation SMB Read

Find File in Workgroup is illustrative of typical User interaction with the File Access Services System as a User locates a file server and a share on it, obtains a directory listing, and then opens a file.

6.1.2.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.

6.1.2.2 Sequence of Events

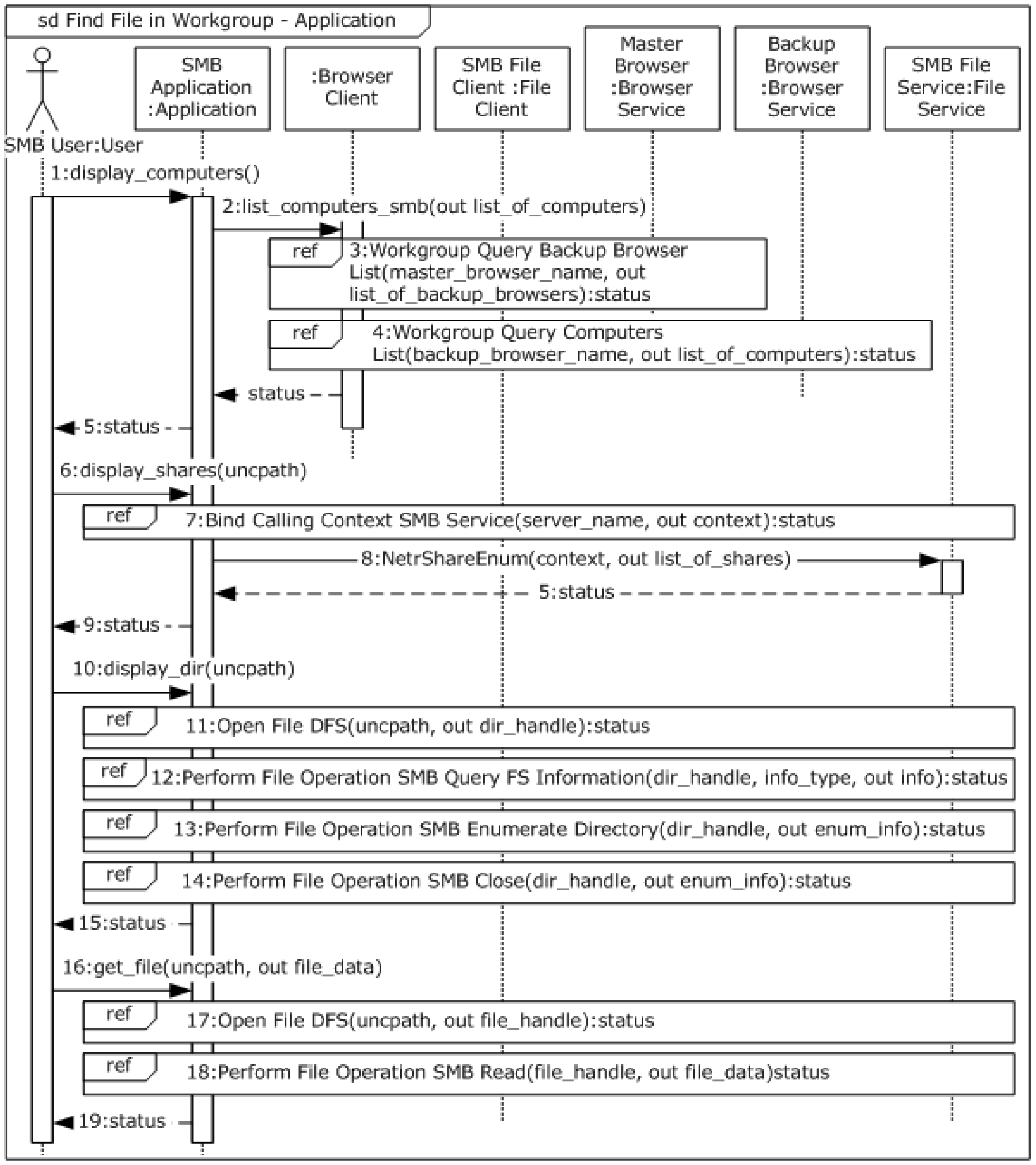

sd Find File in Workgroup - Application
SMB User:User
SMB Application :Application
:Browser Client
SMB File Client :File Client
Master Browser :Browser Service
Backup Browser :Browser Service
SMB File Service:File Service
1:display_computers()
2:list_computers_smb(out list_of_computers)
ref 3:Workgroup Query Backup Browser List(master_browser_name, out list_of_backup_browsers):status
ref 4:Workgroup Query Computers List(backup_browser_name, out list_of_computers):status
status
5:status
6:display_shares(uncpath)
ref 7:Bind Calling Context SMB Service(server_name, out context):status
8:NetrShareEnum(context, out list_of_shares)
5:status
9:status
10:display_dir(uncpath)
ref 11:Open File DFS(uncpath, out dir_handle):status
ref 12:Perform File Operation SMB Query FS Information(dir_handle, info_type, out info):status
ref 13:Perform File Operation SMB Enumerate Directory(dir_handle, out enum_info):status
ref 14:Perform File Operation SMB Close(dir_handle, out enum_info):status
15:status
16:get_file(uncpath, out file_data)
ref 17:Open File DFS(uncpath, out file_handle):status
ref 18:Perform File Operation SMB Read(file_handle, out file_data)status
19:status

Figure 14: Sequence diagram for Find File in Workgroup use case

1. SMB User requests SMB Application to present a display of the list of computers in their workgroup. display_computers() is an example of an application specific message that SMB User uses to direct SMB Application to display a list of computers in the workgroup.

2. SMB Application requests a list of computers from **Browser Client**. list_computers_smb() is an example of an application specific message that the SMB Application initiates to direct Browser Client to retrieve computers in the workgroup.

3. Browser Client makes use of "Workgroup Query Backup Browser List" sequence to retrieve the list of backup **browsers** in the workgroup.

4. Browser Client makes use of "Workgroup Query Computers List" sequence to retrieve the list of computers in the workgroup from Backup Browser.

5. SMB Application presents a display of the list of computers in the workgroup to SMB User.

6. SMB User requests SMB Application to present a display of the list of shares on a file server represented by UNC path. display_shares() is an example of an application specific message that SMB User uses to direct SMB Application to display a list of shares on a file server in the workgroup.

7. SMB Application makes use of "Bind Calling Context SMB Service" sequence to establish a session to SMB File Service on the file server.

8. SMB Application uses **NetrShareEnum** message as described in [MS-SRVS] section 3.1.4.8 to retrieve the list of shares from SMB File Service.

9. SMB Application presents a display of the list of shares on the file server to SMB User.

10. SMB User requests SMB Application to present a display of the contents of a directory represented by UNC path. display_dir() is an example of an application specific message that SMB User uses to direct SMB Application to display the contents of a directory.

11. SMB Application makes use of "Open File DFS" sequence to obtain a handle to the directory represented by **uncpath**.

12. SMB Application makes use of "Perform File Operation SMB Query FS Information" sequence to obtain information such as the label and serial number of the volume hosting the object store backing the SMB file share.

13. SMB Application makes use of "Perform File Operation SMB Enumerate Directory" sequence to enumerate the contents of the directory and their attributes.

14. SMB Application makes use of "Perform File Operation SMB Close" sequence to close the directory handle previously opened.

15. SMB Application presents a display of the contents of the directory to SMB User.

16. SMB User requests SMB Application to present a display of the contents of a specific file of interest in the previous display as represented by uncpath. get_file() is an example of an application specific message that SMB User uses to direct SMB Application to display the contents of a file.

17. SMB Application makes use of "Open File DFS" sequence to obtain a handle to the file represented by uncpath.

18. SMB Application makes use of "Perform File Operation SMB Read" sequence to obtain the contents of the file.

19. SMB Application presents a display of the contents of the file to SMB User.

6.1.2.3 Final System State

The SMB User has obtained a display of a specific file of interest within their workgroup.

6.1.3 Find a File in a Domain

This example illustrates the Find File in Domain summary use case and each of the included detail use cases:

- Open File DFS
- Perform File Operation SMB Query FS Information
- Perform File Operation SMB Enumerate Directory
- Perform File SMB Close
- Perform File Operation SMB Read

Find File in Domain is illustrative of typical User interaction with the File Access Services System as the User locates a file within a DFS namespace and then opens the file.

6.1.3.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- The participating client and server computers must be configured to belong to the same Active Directory domain.
- A DFS domain namespace must be present in the Active Directory domain. This namespace contains at least one link, whose target contains a file which the User will access.
- A Hosted Cache is available, the SMB File Client is configured to use Hosted Cache, and the SMB File Service and SMB File Client both support [MS-SMB2] dialect 2.1.

6.1.3.2 Sequence of Events

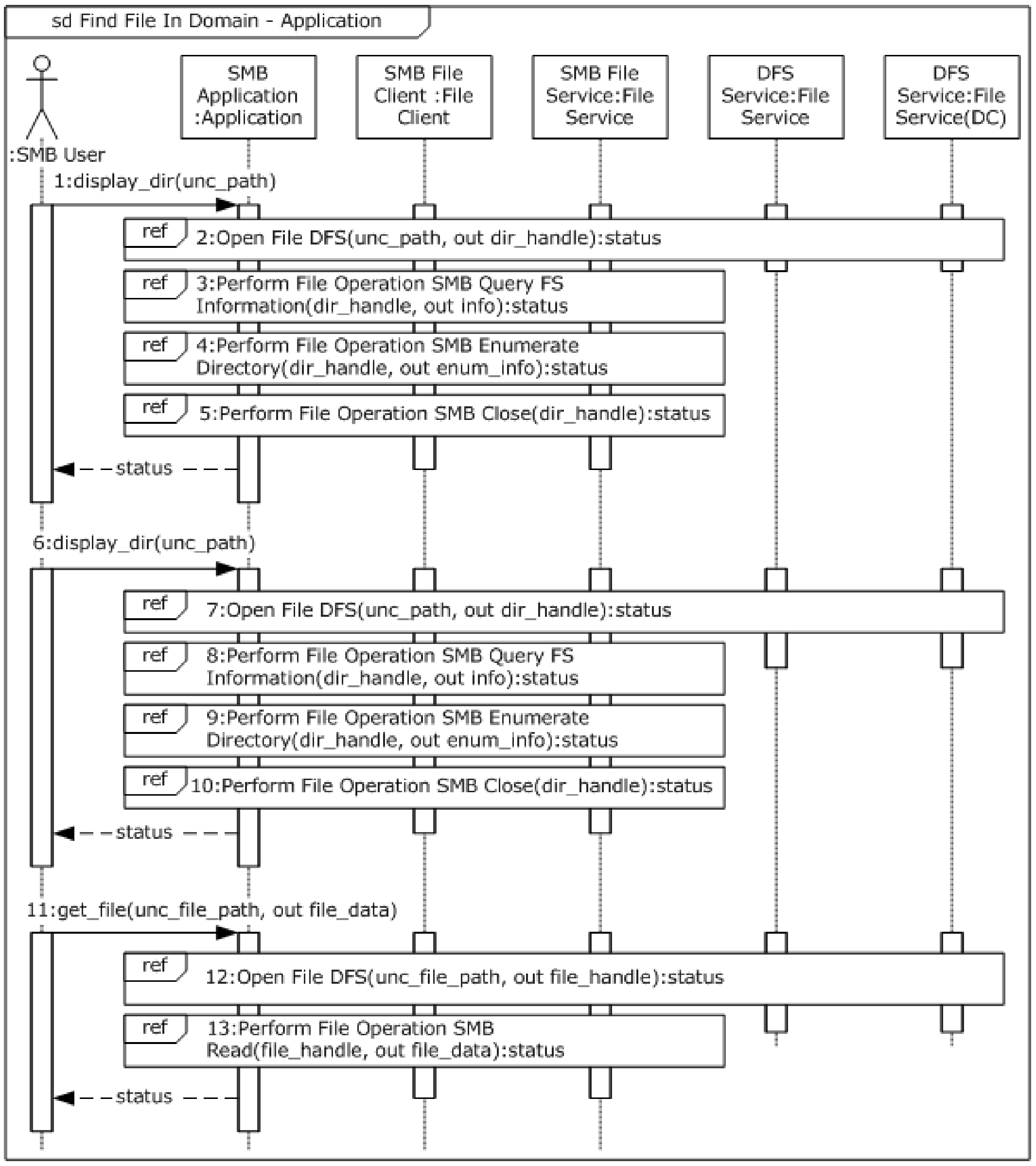

sd Find File In Domain - Application
:SMB User
SMB Application :Application
SMB File Client :File Client
SMB File Service:File Service
DFS Service:File Service
DFS Service:File Service(DC)
1:display_dir(unc_path)
ref 2:Open File DFS(unc_path, out dir_handle):status
ref 3:Perform File Operation SMB Query FS Information(dir_handle, out info):status
ref 4:Perform File Operation SMB Enumerate Directory(dir_handle, out enum_info):status
ref 5:Perform File Operation SMB Close(dir_handle):status
status
6:display_dir(unc_path)
ref 7:Open File DFS(unc_path, out dir_handle):status
ref 8:Perform File Operation SMB Query FS Information(dir_handle, out info):status
ref 9:Perform File Operation SMB Enumerate Directory(dir_handle, out enum_info):status
ref 10:Perform File Operation SMB Close(dir_handle):status
status
11:get_file(unc_file_path, out file_data)
ref 12:Open File DFS(unc_file_path, out file_handle):status
ref 13:Perform File Operation SMB Read(file_handle, out file_data):status
status

Figure 15: Sequence diagram for Find File in Domain use case

1. User requests the SMB Application to present a display of the contents of the directory specified by unc_path, at the root of the DFS domain namespace.

2. The SMB Application makes use of the "Open File DFS" sequence diagram to obtain a handle to the directory, navigating to a root target of the DFS domain namespace.

3. The SMB Application makes use of the "Perform File Operation SMB Query FS Information" sequence diagram to learn about file system attributes, such as available free space, to present to the User.

4. The SMB Application makes use of the "Perform File Operation SMB Enumerate Directory" sequence diagram to discover the names and attributes of the contents of the directory.
5. The SMB Application makes use of the "Perform File Operation SMB Close" sequence diagram to release its handle to the directory, and presents its display to the user.
6. The User requests the SMB Application to present a display of the contents of a specific directory of interest in the previous display. For the purpose of this example, this directory is assumed to be a DFS link.
7. The SMB Application makes use of the "Open File DFS" sequence diagram to obtain a handle to the directory, navigating to a target of the DFS link.
8. Repeating the action of step 3, the SMB Application makes use of the "Perform File Operation SMB Query FS Information" sequence diagram to learn about file system attributes, such as available free space, to present to the User.
9. Repeating the action of step 4, the SMB Application makes use of the "Perform File Operation SMB Enumerate Directory" sequence diagram to discover the names and attributes of the contents of the directory.
10. Repeating the action of step 5, The SMB Application makes use of the "Perform File Operation SMB Close" sequence diagram to release its handle to the directory, and presents its display to the user.
11. The User requests the SMB Application to present a display of the contents of a specific file of interest in the previous display.
12. The SMB Application makes use of the "Open File DFS" sequence diagram to obtain a handle to the file.
13. The SMB Application makes use of the "Perform File Operation SMB Read" sequence diagram to obtain the contents of the file, and presents its display to the user.

6.1.3.3 Final System State

The User has obtained a display of a specific file of interest within the DFS namespace.

6.1.4 Two Applications Communicate using a Shared File

This example illustrates the Communicate through Shared File summary use case and each of the included detail use cases:

- Perform File Operation SMB Request Directory Change Notification,
- Open File NFS
- Perform File Operation NFS Write
- Act On Directory Change Notification SMB
- Open File SMB
- Perform File Operation SMB Read
- Perform File Operation SMB Write
- Perform File Operation SMB Close

- Perform File Operation NFS Read

The sequence described in this example allows two applications, one using the NFS File Access Protocol and another using the SMB File Access Protocol to share a file on a remote file server.

6.1.4.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Participating client and server computers must be configured to belong to the same Active Directory domain.
- The NFS and SMB File shares must be preconfigured on the file server computer to be backed by the same directory in the Object Store and must be provisioned to allow SMB User and NFS User access to the SMB File Share and NFS File Share respectively. Further, the client computer from which the NFS User will access the NFS File Share must be granted access on the NFS File Share.
- The NFS File Service must be configured to use Active Directory with [RFC2307] functionality as the LDAP mapping store for user account mapping.
- The file server hosting the NFS File Share must be configured to do LDAP lookups on the Active Directory.
- The user account corresponding to the SMB User in Active Directory must have the **uidNumber** attribute set to the value of the **user ID** field of the AUTH_SYS credentials corresponding to the NFS User.
- A group account corresponding to the group that the SMB User belongs to in Active Directory must have the **gidNumber** attribute set to the value of the **group ID** field of the AUTH_SYS credentials corresponding to the NFS User.
- The directory in the Object Store on the file server that is being shared by the SMB File Share and NFS File Share must not already have a file with the same name as the file being shared in this example.

6.1.4.2 Sequence of Events

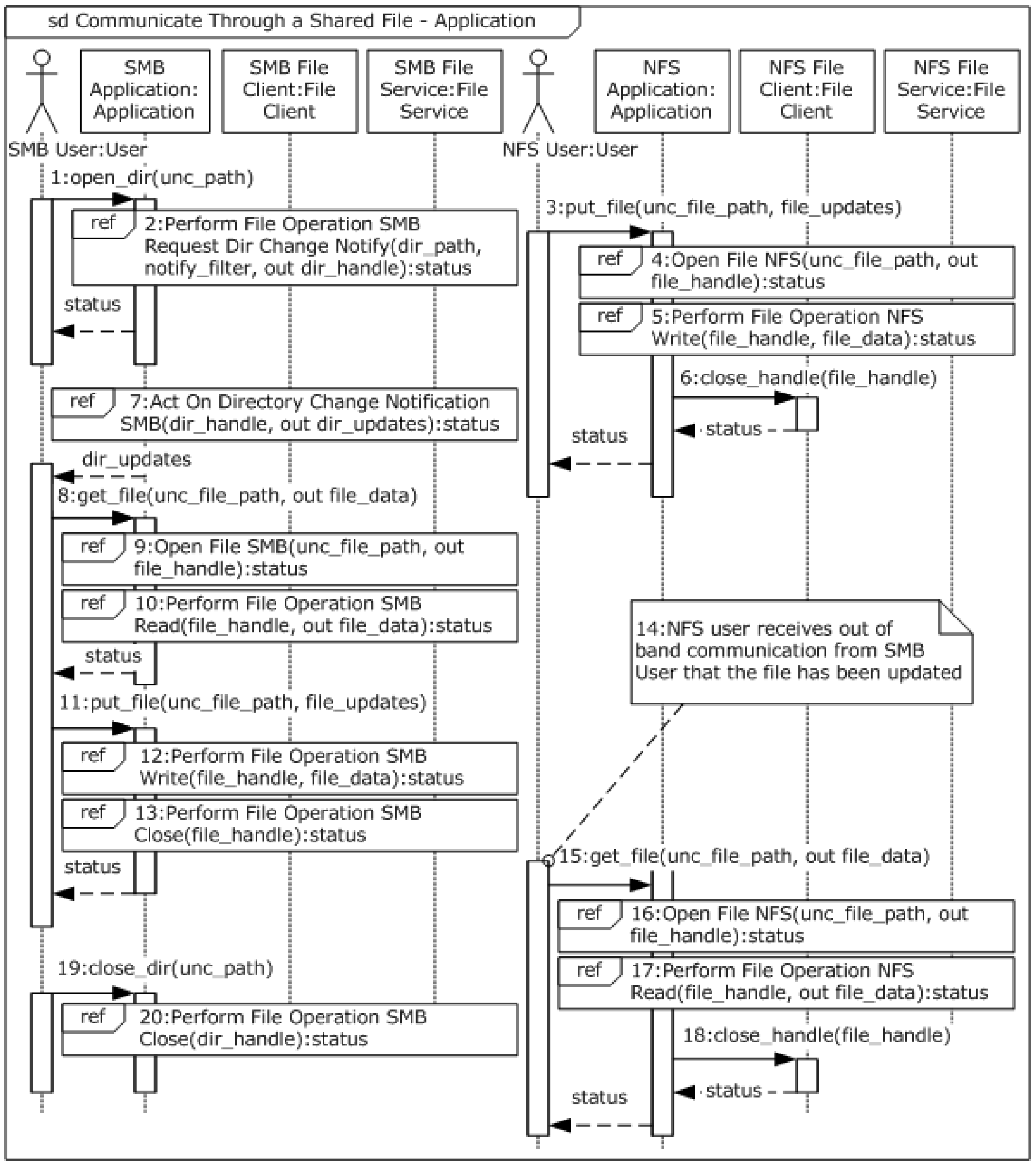

sd Communicate Through a Shared File - Application
SMB User:User
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
NFS User:User
NFS Application: Application
NFS File Client:File Client
NFS File Service:File Service
1:open_dir(unc_path)
ref 2:Perform File Operation SMB Request Dir Change Notify(dir_path, notify_filter, out dir_handle):status
status
3:put_file(unc_file_path, file_updates)
ref 4:Open File NFS(unc_file_path, out file_handle):status
ref 5:Perform File Operation NFS Write(file_handle, file_data):status
6:close_handle(file_handle)
status
status
ref 7:Act On Directory Change Notification SMB(dir_handle, out dir_updates):status
dir_updates
8:get_file(unc_file_path, out file_data)
ref 9:Open File SMB(unc_file_path, out file_handle):status
ref 10:Perform File Operation SMB Read(file_handle, out file_data):status
status
14:NFS user receives out of band communication from SMB User that the file has been updated
11:put_file(unc_file_path, file_updates)
ref 12:Perform File Operation SMB Write(file_handle, file_data):status
ref 13:Perform File Operation SMB Close(file_handle):status
status
15:get_file(unc_file_path, out file_data)
ref 16:Open File NFS(unc_file_path, out file_handle):status
ref 17:Perform File Operation NFS Read(file_handle, out file_data):status
19:close_dir(unc_path)
ref 20:Perform File Operation SMB Close(dir_handle):status
18:close_handle(file_handle)
status
status

Figure 16: Sequence diagram for Communicate through Shared File use case

1. The SMB User navigates to the UNC path of the SMB file share using the SMB Application. open_dir() is an example of an application specific message that the SMB User initiates to direct the SMB Application to communicate with the SMB File Service on a file server.

2. SMB Application extracts the remote file server, the SMB file share and the target directory from the UNC path and uses SMB File Client to establish a session to the SMB File Service. On successful session establishment, the SMB Application uses the SMB File Client to request a directory change notification operation on the opened directory. On successful completion of the request for directory change notification, the SMB File Client returns a directory handle,

dir_handle, to the SMB Application. This sequence of steps is covered in the "Perform File Operation SMB Request Directory Change Notification" sequence diagram.

3. The NFS User instructs the NFS Application to create a file on the NFS file share. put_file() is an example of an application specific message that the NFS User initiates to direct the NFS Application to communicate with the NFS File Service on the file server.

4. The NFS Application extracts the remote File Server, the NFS File Share and the path to the new file to create from the unc_file_path, and uses the NFS File Client to create the file on the NFS file share. On successful completion of this step, NFS Application receives a handle, file_handle, corresponding to the newly created file from the NFS File Client. This sequence of steps is covered in the "Open File NFS" sequence diagram.

5. On successful completion of the previous step, the NFS Application writes the user supplied data to the newly created file. This is covered in the "Perform File Operation NFS Write" sequence diagram.

6. The NFS Application closes the previously opened file handle by calling close_handle() in the NFS File Client.

7. The SMB File Service gets notifications from the Object Store indicating that a new file has been created, data has been written and the file size has changed. In response, the SMB File Service sends a notification message to the SMB File Client which had previously registered for the directory change notification. On receipt of this notification, the SMB File Client in turn signals the file handle previously opened by the SMB Application (dir_handle) causing the SMB Application to read the change notification details (dir_updates) and display or notify them to the SMB User. This sequence of steps is described in the "Act On Directory Change Notification SMB" sequence diagram.

8. The SMB User on receipt of the directory change notification discovers that a new file has been created by the NFS User and requests the SMB Application to retrieve the file contents. get_file() is an example of an application specific message that the SMB User initiates to direct the SMB Application to retrieve the file contents from the SMB File Share on the file server.

9. The SMB Application uses the SMB File Client to open the requested file on the SMB File Share. As a result of a successful open, the SMB File Client returns a handle to the opened file, file_handle, to the SMB Application. This sequence of steps is covered in the "Open File DFS" sequence diagram.

10. The SMB Application then initiates a read operation using the SMB File Client to retrieve the contents of the file opened in the previous step and display or notify them to the SMB User. This sequence of steps is covered in the "Perform File Operation SMB Read" sequence diagram.

11. The SMB User updates the contents of the file. put_file() is an example of an application specific message that the SMB User initiates to instruct the SMB Application to write new contents into the file.

12. The SMB Application uses the previously opened file handle, file_handle, to request the SMB File Client to update the file using the user supplied contents, file_data. This sequence of steps is covered in the "Perform File Operation SMB Write" sequence diagram.

13. The SMB Application closes the file handle, file_handle as described in the "Perform File Operation SMB Close" sequence diagram.

14. The NFS User is notified that the file has been updated by the SMB User. The notification mechanism is not a part of this sequence diagram and could be one of a variety of mechanisms

that both users use to communicate. For example, the SMB User could send an email to the NFS User or use the telephone to notify that the file is ready to be viewed by the NFS User.

15. The NFS User directs the NFS Application to read the contents of the file on the NFS File Share. get_file() is an example of an application specific message that the NFS User initiates to request the file contents from the NFS Application.

16. The NFS Application extracts the location and obtains a handle to the file to be opened from the NFS File Client. The result of a successful open is a handle, file_handle, returned to the NFS Application from the NFS File Client. This sequence of steps is covered in the "Open File NFS" sequence diagram.

17. The NFS Application retrieves the contents of the file, file_data, as described in the "Perform File Operation NFS Read" sequence diagram.

18. The NFS Application requests a close on the file_handle using the close_handle() message which is an example of an application specific message type implemented by the NFS File Client.

19. The SMB User closes the previously opened directory on the SMB File Share. close_dir() is an example of an application specific message that the SMB User initiates to direct the SMB Application to remove all state previously established for that directory.

20. The SMB Application closes the previously opened directory handle, dir_handle, with the SMB File Client. This sequence of steps is described in the "Perform File Operation SMB Close" sequence diagram.

6.1.4.3 Final System State

A new file has been created in the directory on the Object Store that backs the NFS File Share and SMB File Share. This file contains the data from both the NFS User and SMB User.

6.1.5 Sequence Diagram Details

This section contains details of the sequence diagrams referred to in sections 6.1.1 through 6.1.4.

6.1.5.1 Act On Directory Change Notification SMB

The sequence described in this example details how the SMB Application receives change notifications in response to a previously registered change notification request on a directory in the SMB File Share.

6.1.5.1.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.

- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

- A previous change notification request from the SMB Application at the SMB File Client and the SMB File Client at the SMB File Service as covered by the sequence Perform File Operation SMB Request Directory Change Notification must have completed.

6.1.5.1.2 Sequence of Events

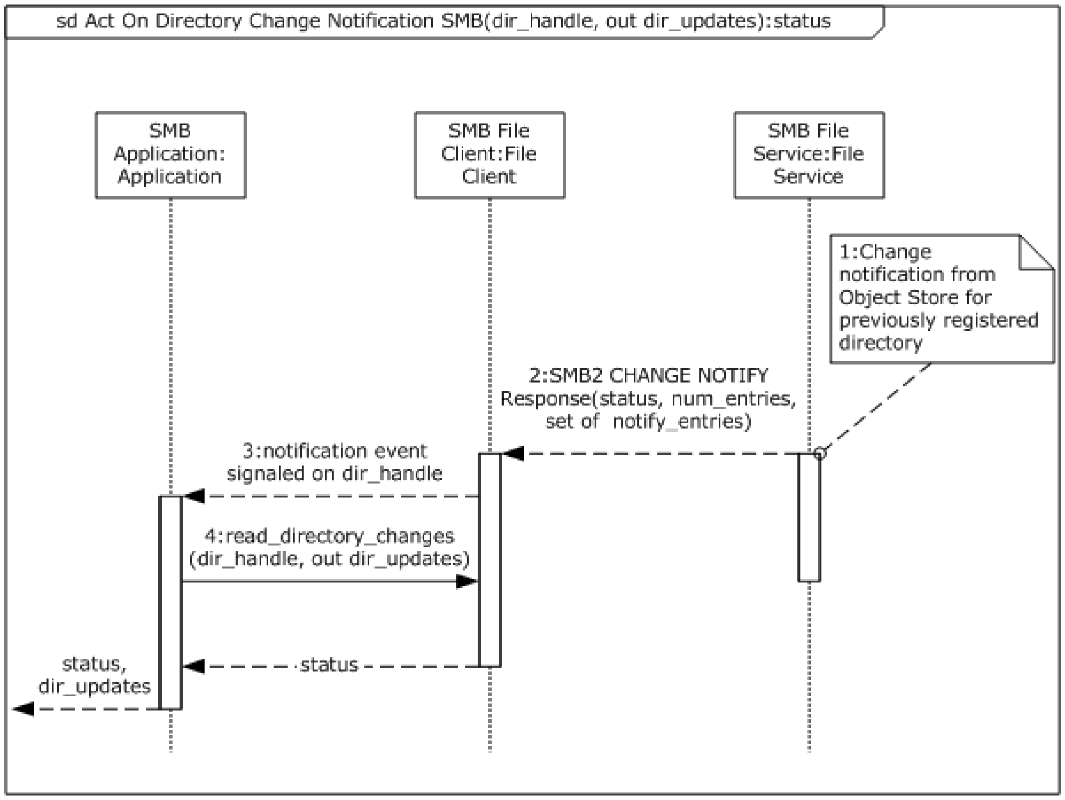

sd Act On Directory Change Notification SMB(dir_handle, out dir_updates):status
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
1:Change notification from Object Store for previously registered directory
2:SMB2 CHANGE NOTIFY Response(status, num_entries, set of notify_entries)
3:notification event signaled on dir_handle
4:read_directory_changes (dir_handle, out dir_updates)
status
status, dir_updates

Figure 17: Sequence diagram detail for Act On Directory Change Notification SMB

1. The SMB File Service receives a notification from the underlying object store in response to changes in the directory that was previously registered for notifications. The mechanism of this notification is Object Store implementation specific and is not covered in this document.

2. In response to the change notification from the Object Store, the SMB File Service sends an SMB2 CHANGE NOTIFY Response message, [MS-SMB2] section 2.2.36, to the SMB File Client. As a part of this message the SMB File Client receives details of the changes that occurred in the directory namespace as represented by the set of notify_entries. This is described further in [MS-SMB2] section 3.3.5.19.

3. The SMB File Client indicates the receipt of notification event from the SMB File Service by signaling the dir_handle that was returned previously on successful completion of "Perform File Operation SMB Request Directory Change Notification" sequence.

4. The SMB Application next reads the changes that occurred in the directory representing dir_handle by sending the read_directory_changes() message to the SMB File Client. read_directory_changes() is an example of an application specific message that the SMB Application uses to read the set of changes in the directory. On successful completion of this message, the SMB Application receives the updates that happened in the directory as represented by dir_updates.

6.1.5.1.3 Final System State

The SMB Application receives the set of changes that occurred in the directory on the SMB File Share.

6.1.5.2 Bind Calling Context DFS Service

The sequence described in this example details how the Application establishes an RPC calling context for the purpose of issuing methods defined in [MS-DFSNM] to a DFS Service.

6.1.5.2.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.2.2 Sequence of Events

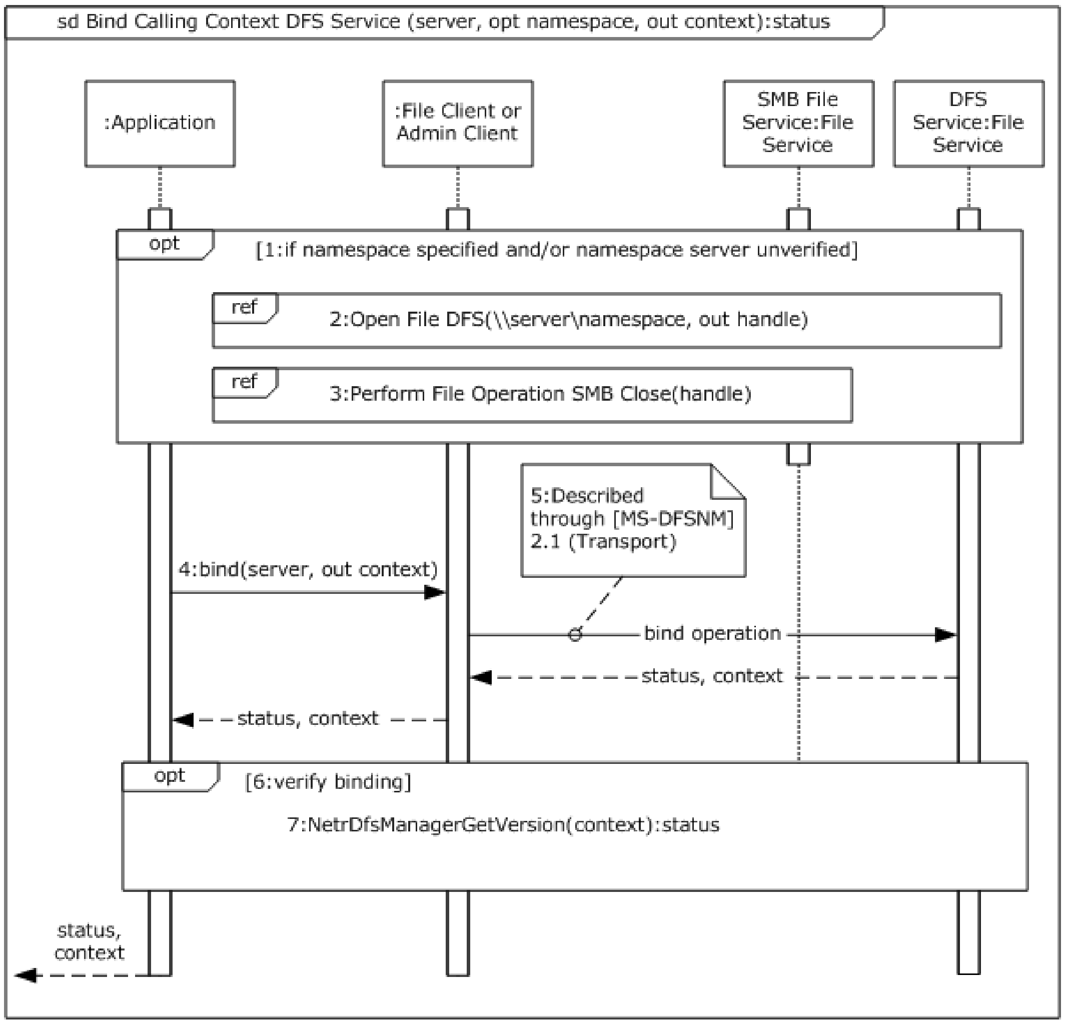

sd Bind Calling Context DFS Service (server, opt namespace, out context):status
:Application
:File Client or Admin Client
SMB File Service:File Service
DFS Service:File Service
opt
[1:if namespace specified and/or namespace server unverified]
ref
2:Open File DFS(\\server\namespace, out handle)
ref
3:Perform File Operation SMB Close(handle)
5:Described through [MS-DFSNM] 2.1 (Transport)
4:bind(server, out context)
bind operation
status, context
status, context
opt
[6:verify binding]
7:NetrDfsManagerGetVersion(context):status
status, context

Figure 18: Sequence diagram detail for Bind Calling Context DFS Service

1. The Application acquires a server to bind the RPC calling context to. If the caller specifies the binding in the form of a namespace ("\\server\namespace" or "\\domain\namespace"), the Application consults the ReferralCache, described in the [MS-DFSC] 3.1.1 Abstract Data Model, of the File or Admin Client. If the ReferralCache entry for the specified namespace is available and identifies an available root target server, the Application continues operation at step 4 using this server.

2. The Application performs an open using the sequence "Open File DFS", which identifies an available root target server of the namespace if successful.

3. The Application closes the handle acquired in step 2 using the sequence Perform File Operation SMB Close.

4. The Application uses the identified server to request the File or Admin Client to bind the RPC calling context.

5. The File or Admin Client establishes the calling context using the procedure specified in [MS-DFSNM] 2.1 Transport.

6. The Application may, at its discretion, verify the calling context using the [MS-DFSNM] 3.1.4.1.2 **NetrDfsManagerGetVersion** method. This choice does not affect the operation of the system.

6.1.5.2.3 Final System State

- The Application has an RPC calling context it can use for the purpose of issuing methods defined in [MS-DFSNM].

- The File or Admin Client has state established on the DFS Service represented by the context returned.

6.1.5.3 Bind Calling Context FSRM Service

The sequence described in this example details how the Application establishes an RPC calling context for the purpose of issuing methods defined in [MS-FSRM] to an FSRM Service.

6.1.5.3.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.

- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.3.2 Sequence of Events

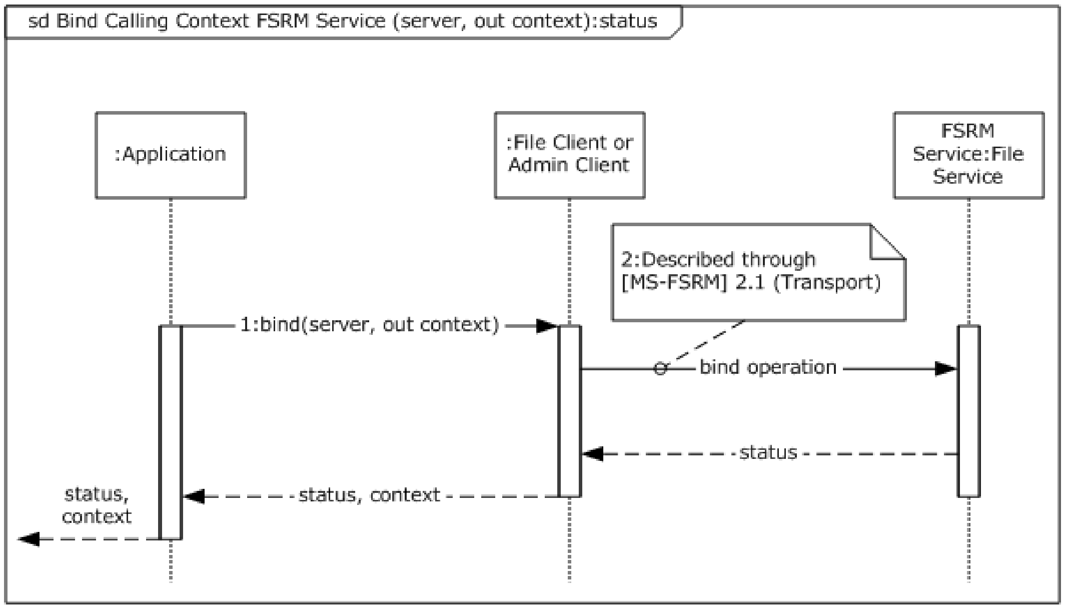

sd Bind Calling Context FSRM Service (server, out context):status
:Application
:File Client or Admin Client
FSRM Service:File Service
2:Described through [MS-FSRM] 2.1 (Transport)
1:bind(server, out context)
bind operation
status
status, context
status, context

Figure 19: Sequence diagram detail for Bind Calling Context FSRM Service

1. The Application uses the identified server to request the File or Admin Client to bind the RPC calling context.
2. The File or Admin Client establishes the calling context using the procedure specified in [MS-FSRM] 2.1 Transport.

6.1.5.3.3 Final System State

- The Application has an RPC calling context it can use for the purpose of issuing methods defined in [MS-FSRM].
- The File or Admin Client has state established on the FSRM Service represented by the context returned.

6.1.5.4 Bind Calling Context SMB Service

The sequence described in this example details how the Application establishes an RPC calling context for the purpose of issuing methods defined in [MS-SRVS] to an SMB Service.

6.1.5.4.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.4.2 Sequence of Events

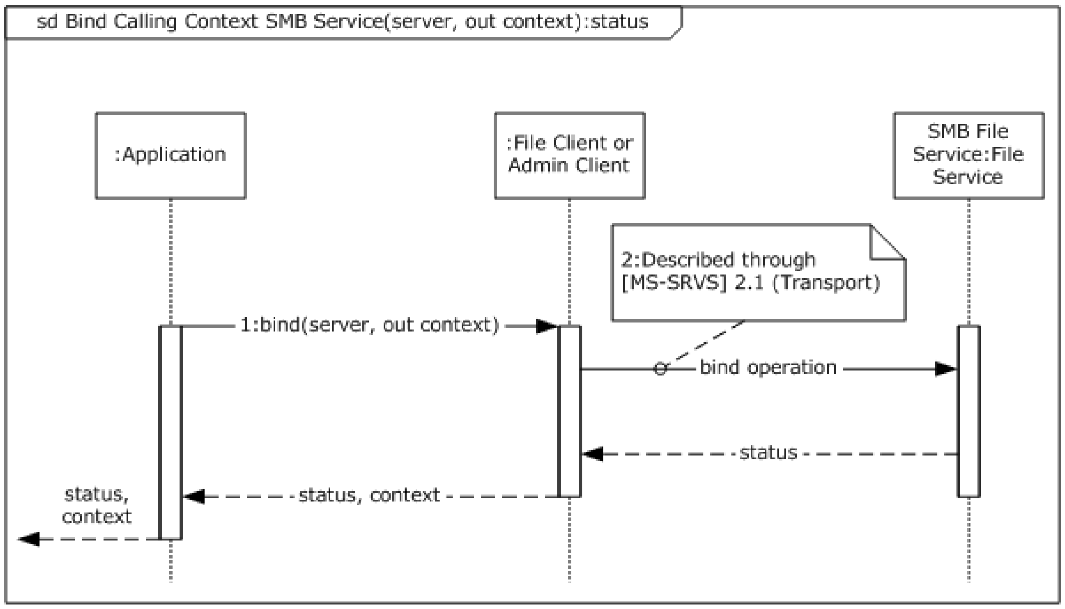

sd Bind Calling Context SMB Service(server, out context):status
:Application
:File Client or Admin Client
SMB File Service:File Service
2:Described through [MS-SRVS] 2.1 (Transport)
1:bind(server, out context)
bind operation
status
status, context
status, context

Figure 20: Sequence diagram detail for Bind Calling Context SMB Service

1. The Application uses the identified server to request the File or Admin Client to bind the RPC calling context.
2. The File or Admin Client establishes the calling context using the procedure specified in [MS-SRVS] 2.1 Transport.

6.1.5.4.3 Final System State

- The Application has an RPC calling context it can use for the purpose of issuing methods defined in [MS-SRVS].
- The File or Admin Client has state established on the SMB Service represented by the context returned.

6.1.5.5 Create DFS Link

The sequence described in this example details how the Application creates a DFS link in a given DFS namespace.

6.1.5.5.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.5.2 Sequence of Events

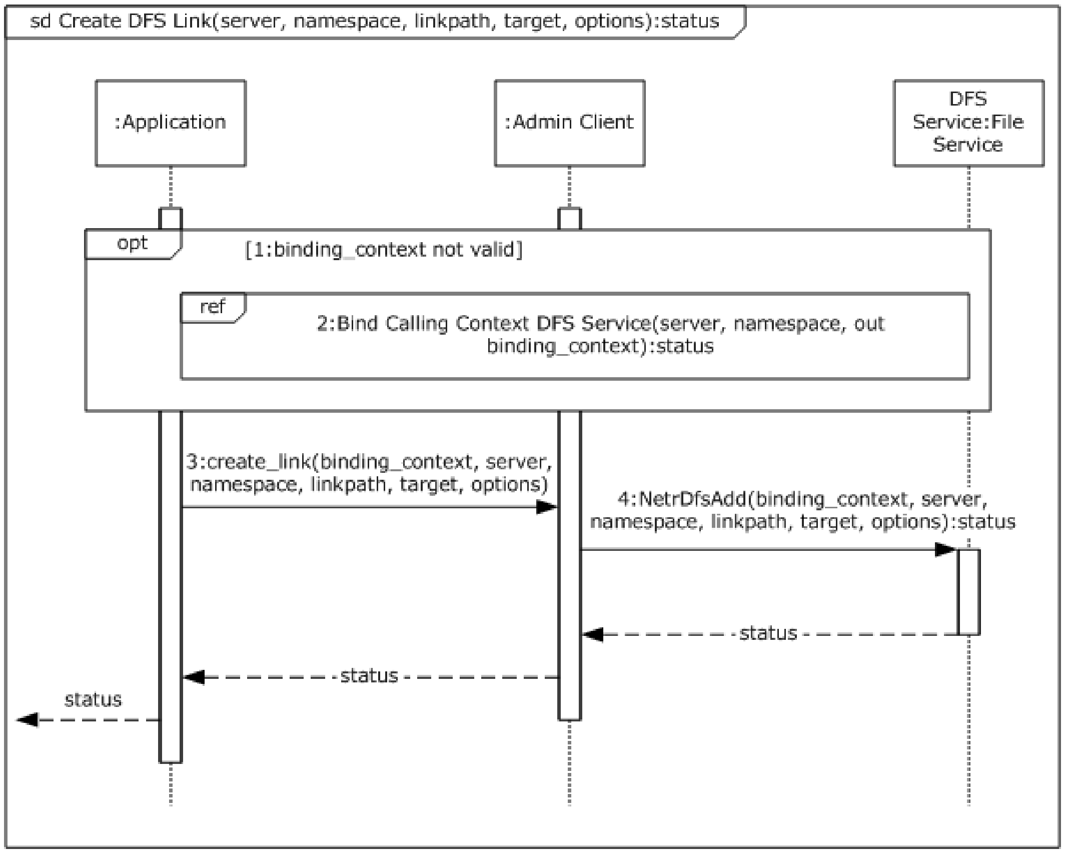

sd Create DFS Link(server, namespace, linkpath, target, options):status
:Application
:Admin Client
DFS Service:File Service
opt
[1:binding_context not valid]
ref
2:Bind Calling Context DFS Service(server, namespace, out binding_context):status
3:create_link(binding_context, server, namespace, linkpath, target, options)
4:NetrDfsAdd(binding_context, server, namespace, linkpath, target, options):status
status
status
status

Figure 21: Sequence diagram detail for Create DFS Link

1. The Application checks to see if it has a valid RPC calling context available to the specified DFS namespace. If so, it continues operation at step 3.
2. The Application acquires an RPC calling context using the sequence described in Bind Calling Context DFS Service, providing the specified server and namespace.
3. The Application requests the Admin Client to create the given DFS link within the DFS namespace hosted by the server identified in the RPC calling context.
4. The Admin Client issues the [MS-DFSNM] 3.1.4.1.3 **NetrDfsAdd** method, specifying the linkpath, target, and various options provided by the use case caller.

6.1.5.5.3 Final System State

- The DFS Service successfully executes the requested operation.
- If the operation is successful, the specified DFS link is created in the DFS namespace hosted by the DFS service.

6.1.5.6 Create DFS Namespace

The sequence described in this example details how the Application creates a DFS namespace on a given server.

6.1.5.6.1 Initial System State

- General requirements as set forth in section 4.2 System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.6.2 Sequence of Events

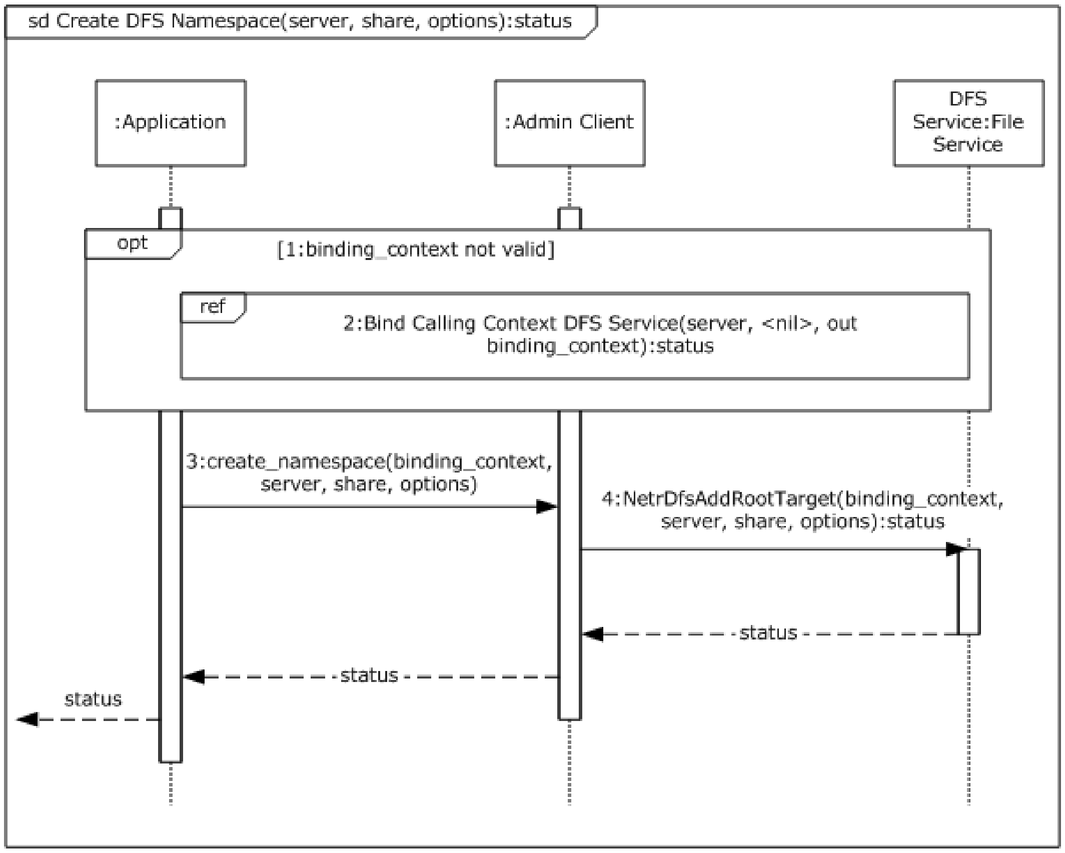

sd Create DFS Namespace(server, share, options):status
:Application
:Admin Client
DFS Service:File Service
opt
[1:binding_context not valid]
ref
2:Bind Calling Context DFS Service(server, <nil>, out binding_context):status
3:create_namespace(binding_context, server, share, options)
4:NetrDfsAddRootTarget(binding_context, server, share, options):status
status
status
status

Figure 22: Sequence diagram detail for Create DFS Namespace

1. The Application checks to see if it has a valid RPC calling context available to the specified server's DFS Service. If so, it continues operation at step 3.
2. The Application acquires an RPC calling context using the sequence described in Bind Calling Context DFS Service, providing the specified server.
3. The Application requests the Admin Client to create the given DFS namespace on the server identified in the RPC calling context.

4. The Admin Client issues the [MS-DFSNM] 3.1.4.1.3 **NetrDfsAddRootTarget** method, specifying the server, share which should host the namespace, and various options provided by the use case caller.

6.1.5.6.3 Final System State

- The DFS Service successfully executes the requested operation.
- If the operation is successful, the specified DFS namespace is created on the server.

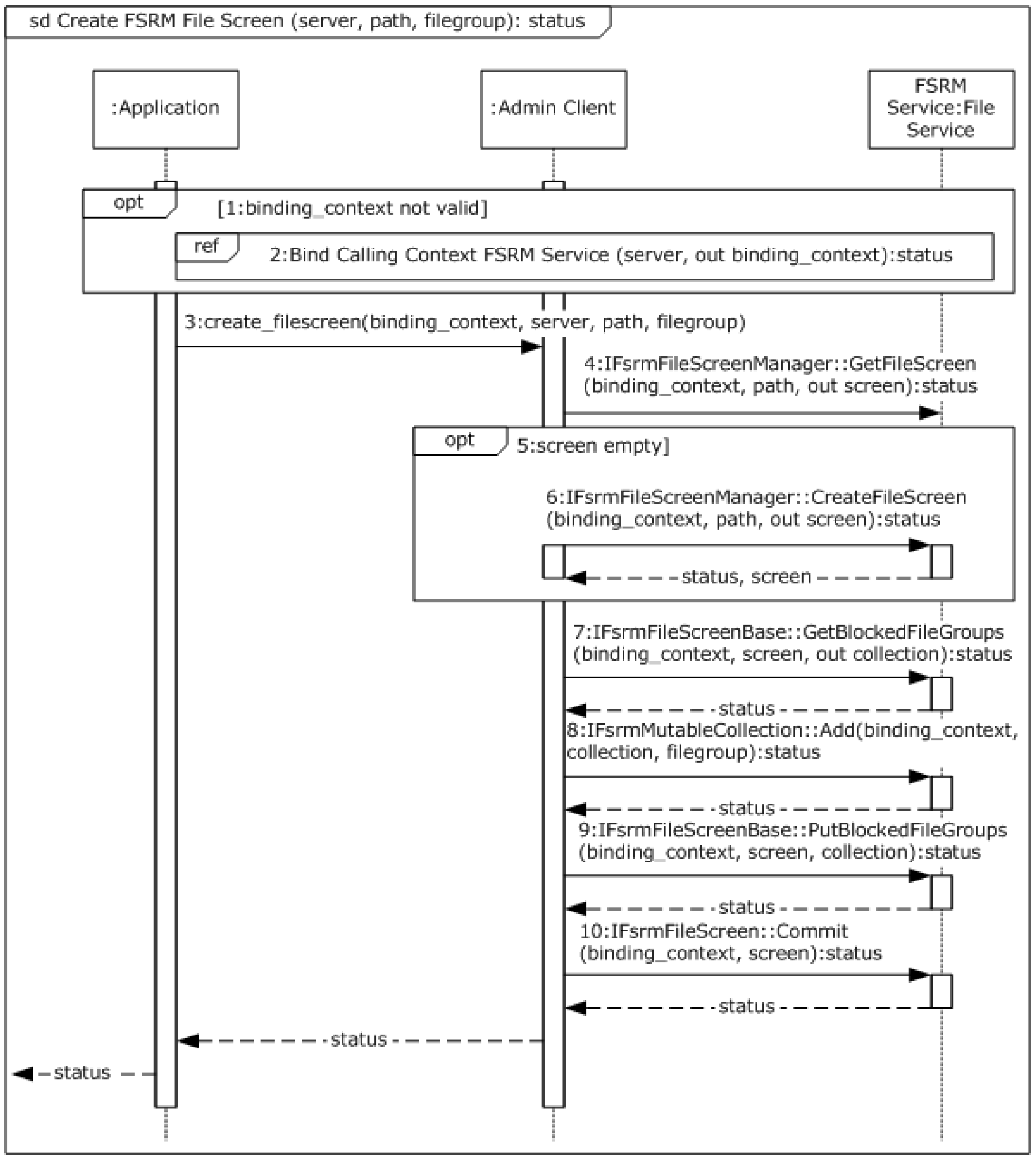
6.1.5.7 Create FSRM File Screen

The sequence described in this example details how the Application creates an FSRM File Screen at a given path on the Object Store of a given server.

6.1.5.7.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.7.2 Sequence of Events



sd Create FSRM File Screen (server, path, filegroup): status
:Application
:Admin Client
FSRM Service:File Service
opt
[1:binding_context not valid]
ref
2:Bind Calling Context FSRM Service (server, out binding_context):status
3:create_filescreen(binding_context, server, path, filegroup)
4:IFsrmFileScreenManager::GetFileScreen (binding_context, path, out screen):status
opt
5:screen empty]
6:IFsrmFileScreenManager::CreateFileScreen (binding_context, path, out screen):status
status, screen
7:IFsrmFileScreenBase::GetBlockedFileGroups (binding_context, screen, out collection):status
status
8:IFsrmMutableCollection::Add(binding_context, collection, filegroup):status
status
9:IFsrmFileScreenBase::PutBlockedFileGroups (binding_context, screen, collection):status
status
10:IFsrmFileScreen::Commit (binding_context, screen):status
status
status
status

Figure 23: Sequence diagram detail for Create FSRM File Screen

1. The Application checks to see if it has a valid RPC calling context available to the specified server's FSRM Service. If so, it continues operation at step 3.

2. The Application acquires an RPC calling context using the sequence described in Bind Calling Context FSRM Service (section 6.1.5.3), providing the specified server.

3. The Application requests the Admin Client to create the given file screen, specifying the RPC calling context, server, local object store path, and file group.

4. The Admin Client queries the server to determine if there is an existing file screen specified on the object store path. To do this, it issues the **IFsrmFileScreenManager::GetFileScreen** method ([MS-FSRM] section 3.2.4.2.29.4), retrieving a potentially nonempty existing **IFsrmFileScreen** object.

5. If an **IFsrmFileScreen** object is returned, the Admin Client has determined that a file screen is currently configured on the server at the specified path and continues operation at step 7.

6. The Admin Client creates an empty **IFsrmFileScreen** object using the [MS-FSRM] 3.2.4.2.29.3 **IFsrmFileScreenManager::CreateFileScreen** method.

7. The Admin Client acquires the **IFsrmMutableCollection** object from the file screen, into which it will add the caller-specified filegroup, using the [MS-FSRM] 3.2.4.2.26.1 **IFsrmFileScreenBase::GetBlockedFileGroups** method.

8. The Admin Client, using the acquired collection object, adds the requested filegroup to the collection using the [MS-FSRM] 3.2.4.2.2.1 **IFsrmMutableCollection::Add** method.

9. The Admin Client places the modified collection object into the file screen, using the [MS-FSRM] 3.2.4.2.26.2 **IFsrmFileScreenBase::PutBlockedFileGroups** method.

10. To complete the operation, the Admin Client instructs the server to commit the modifications to the file screen using the [MS-FSRM] 3.2.4.2.27.1 IFsrmFileScreen::Commit method.

6.1.5.7.3 Final System State

- The FSRM Service successfully executes the requested operations.
- If the operation sequence is successful, the specified file screen policy is created on the server.

6.1.5.8 Create FSRM Quota

The sequence described in this example details how the Application creates a FSRM Quota at a given path on the Object Store of a given server.

6.1.5.8.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.8.2 Sequence of Events

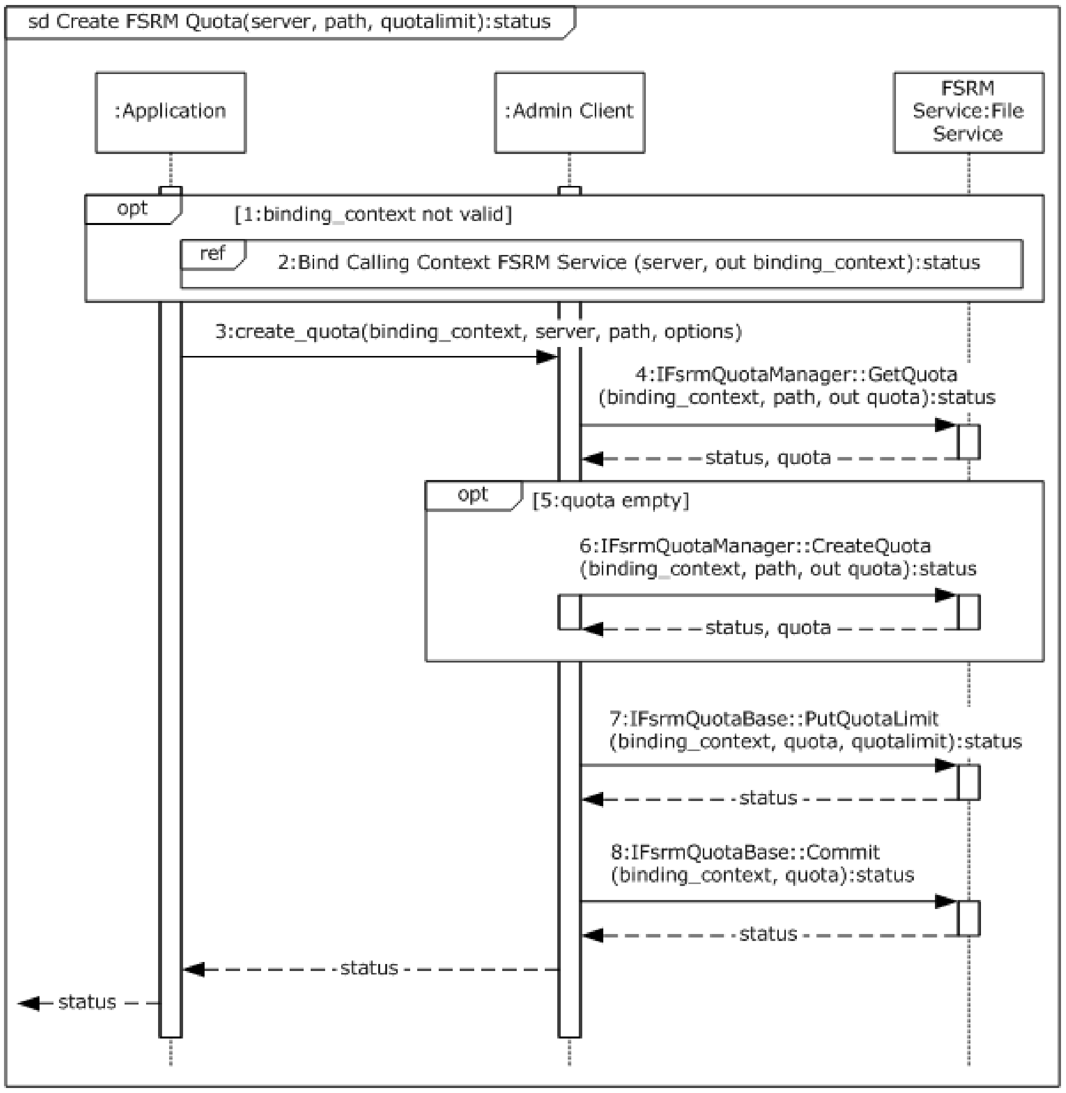

sd Create FSRM Quota(server, path, quotalimit):status
:Application
:Admin Client
FSRM Service:File Service
opt
[1:binding_context not valid]
ref
2:Bind Calling Context FSRM Service (server, out binding_context):status
3:create_quota(binding_context, server, path, options)
4:IFsrmQuotaManager::GetQuota (binding_context, path, out quota):status
status, quota
opt
[5:quota empty]
6:IFsrmQuotaManager::CreateQuota (binding_context, path, out quota):status
status, quota
7:IFsrmQuotaBase::PutQuotaLimit (binding_context, quota, quotalimit):status
status
8:IFsrmQuotaBase::Commit (binding_context, quota):status
status
status
status

Figure 24: Sequence diagram detail for Create FSRM Quota

1. The Application checks to see if it has a valid RPC calling context available to the specified server's FSRM Service. If so, it continues operation at step 3.

2. The Application acquires an RPC calling context using the sequence described in Bind Calling Context FSRM Service, providing the specified server.

3. The Application requests the Admin Client to create the given quota, specifying the RPC calling context, server, local object store path, and quota limit.

4. The Admin Client queries the server to determine if there is an existing quota specified on the object store path. To do this, it issues the [MS-FSRM] 3.2.4.2.18.5 **IFsrmQuotaManager::GetQuota** method, retrieving a potentially non-empty existing **IFsrmQuota** object.

5. If an **IFsrmQuota** object is returned, the Admin Client has determined that quota is currently configured on the server at the specified path and continues operation at step 7.
6. The Admin Client creates an empty **IFsrmQuota** object using the [MS-FSRM] 3.2.4.2.18.3 **IFsrmQuotaManager::CreateQuota** method.
7. The Admin Client modifies the returned **IFsrmQuota** object to reflect the specified quota limit, using the [MS-FSRM] 3.2.4.2.18.3 **IFsrmQuotaBase::PutQuotaLimit** method.
8. To complete the operation, the Admin Client instructs the server to commit the modifications to the quota, using the [MS-FSRM] 3.2.4.2.14.1 IFsrmQuotaBase::Commit method.

6.1.5.8.3 Final System State

- The FSRM Service successfully executes the requested operations.
- If the operation sequence is successful, the specified quota policy is created on the server.

6.1.5.9 Create Share SMB

The sequence described in this example details how the Application creates an SMB share at a given path on the Object Store of a given server.

6.1.5.9.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.9.2 Sequence of Events

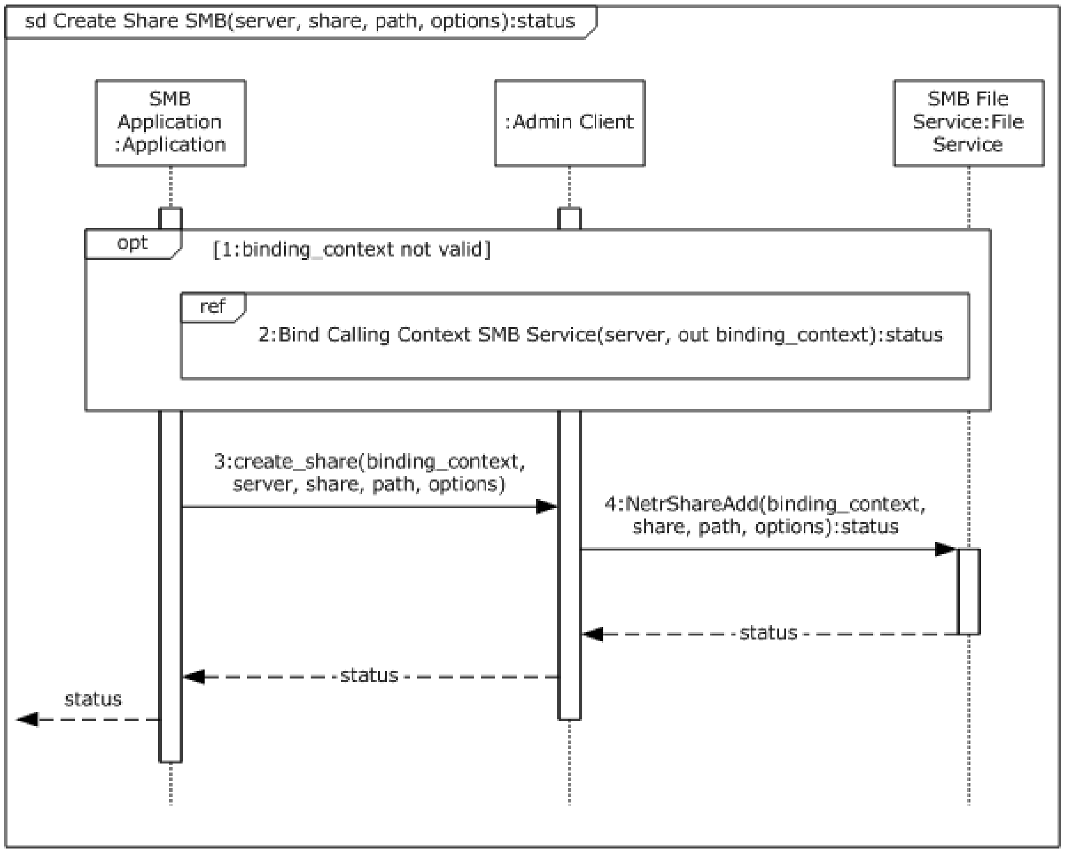

sd Create Share SMB(server, share, path, options):status
SMB Application :Application
:Admin Client
SMB File Service:File Service
opt
[1:binding_context not valid]
ref
2:Bind Calling Context SMB Service(server, out binding_context):status
3:create_share(binding_context, server, share, path, options)
4:NetrShareAdd(binding_context, share, path, options):status
status
status
status

Figure 25: Sequence diagram detail for Create Share SMB

1. The Application checks to see if it has a valid RPC calling context available to the specified SMB File Service. If so, it continues operation at step 3.
2. The Application acquires an RPC calling context using the sequence described in Bind Calling Context SMB Service, providing the specified server.
3. The Application requests the Admin Client to create the given SMB share at the specified path on the local object store on the server identified in the RPC calling context, with various options.
4. The Admin Client issues the [MS-SRVS] 3.1.4.7 **NetrShareAdd** method, specifying the share name, local object store path, and various options provided by the use case caller.

6.1.5.9.3 Final System State

- The SMB Service successfully executes the requested operation.
- If the operation is successful, the specified SMB share is created.

6.1.5.10 Mount Share NFS

The sequence described in this example shows how the NFS File Client obtains the initial NFS file handle for the NFS File Share.

6.1.5.10.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.
- The NFS File Client does not already have a dir_handle that corresponds to NFS File Share to NFS File Handle mapping for that share path on the file server.

6.1.5.10.2 Sequence of Events

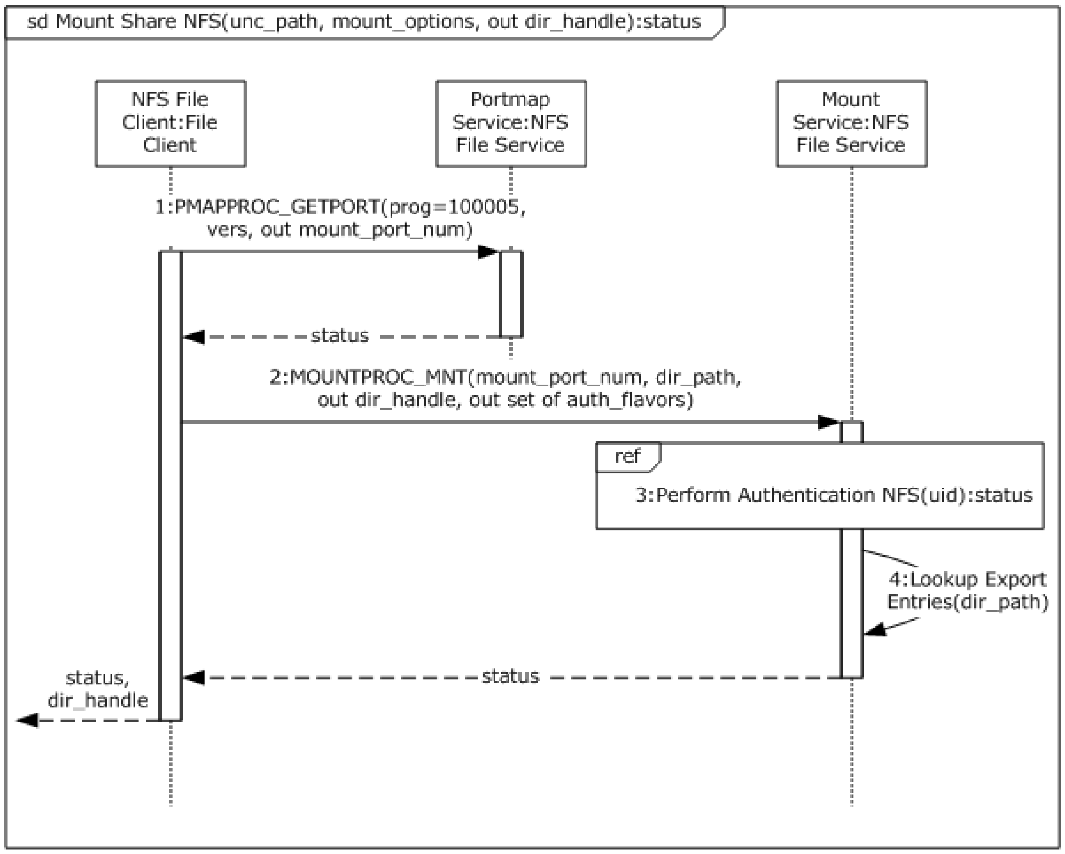

sd Mount Share NFS(unc_path, mount_options, out dir_handle):status
NFS File Client:File Client
Portmap Service:NFS File Service
Mount Service:NFS File Service
1:PMAPPROC_GETPORT(prog=100005, vers, out mount_port_num)
status
2:MOUNTPROC_MNT(mount_port_num, dir_path, out dir_handle, out set of auth_flavors)
ref
3:Perform Authentication NFS(uid):status
4:Lookup Export Entries(dir_path)
status
status, dir_handle

Figure 26: Sequence diagram detail for Mount Share NFS

1. The NFS File Client parses the unc_path to the file server and extracts the server name, NFS Share and the file path. It then sends a PMAPPROC_GETPORT request to the Portmap Service as described in [RFC1833], section 3.2 resulting in the port number for the Mount Service on the file server.
2. The NFS File Client then contacts the Mount File Service on the file server with the MOUNTPROC_MNT request as described in [RFC1813], section 5.2.1 to get the NFS File Handle corresponding to the NFS Share name.
3. The Mount Service on the file server authenticates the user identity as described in the sequence Perform NFS Authentication (section 6.1.5.23).

4. On successful completion of authentication in the previous step, the Mount Service validates the directory path, dir_path, that the NFS File Client sent is provisioned for NFS sharing on the file server, as indicated by the message "Lookup Export Entries". On successful match of the dir_path, the Mount Service returns the NFS handle corresponding to the dir_path.

6.1.5.10.3 Final System State

The NFS File Client successfully mounts the NFS File Share on the file server.

6.1.5.11 Open File DFS

The sequence described in this example details how the SMB Application uses the SMB File Client to open a file on the SMB File Share using mechanisms appropriate to paths which may encounter DFS namespaces.

6.1.5.11.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.11.2 Sequence of Events

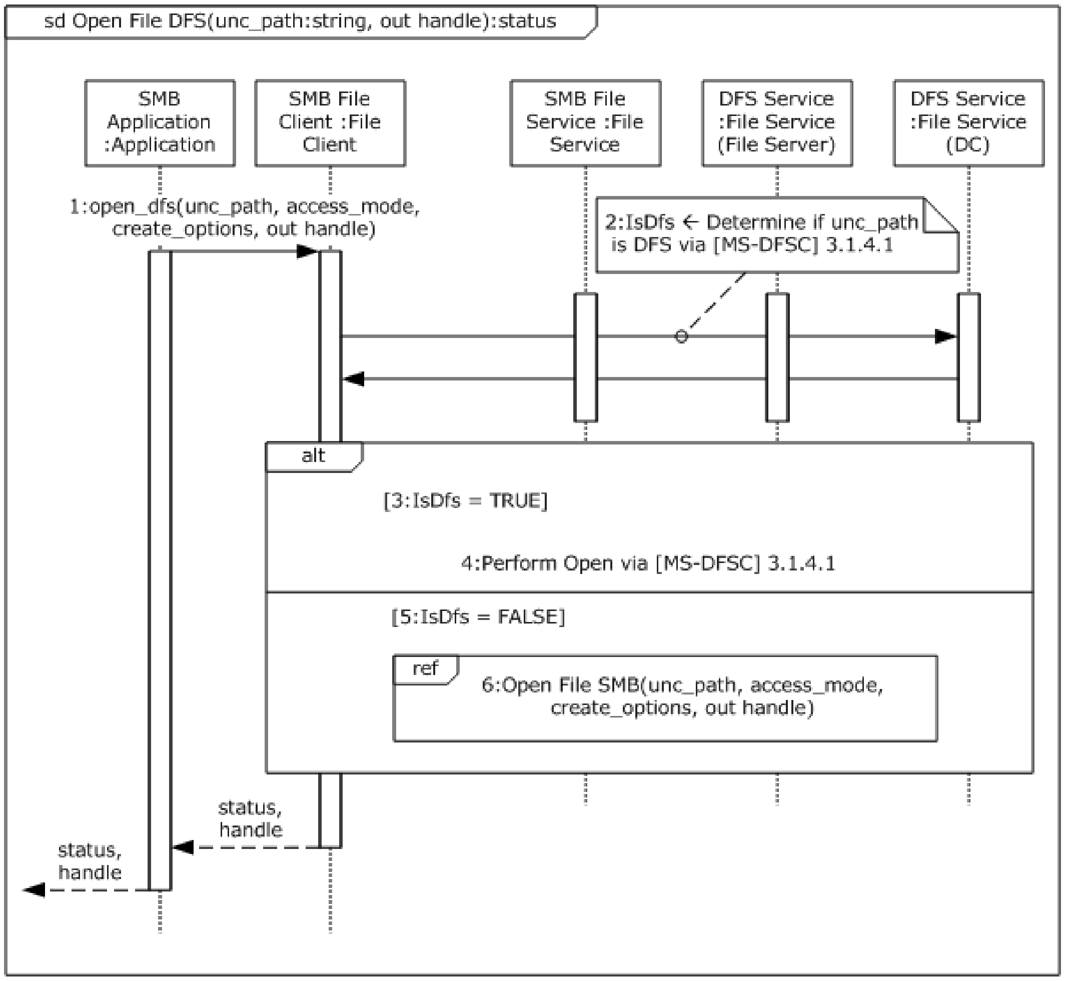

sd Open File DFS(unc_path:string, out handle):status
SMB Application :Application
SMB File Client :File Client
SMB File Service :File Service
DFS Service :File Service (File Server)
DFS Service :File Service (DC)
1:open_dfs(unc_path, access_mode, create_options, out handle)
2:IsDfs ← Determine if unc_path is DFS via [MS-DFSC] 3.1.4.1
alt
[3:IsDfs = TRUE]
4:Perform Open via [MS-DFSC] 3.1.4.1
[5:IsDfs = FALSE]
ref
6:Open File SMB(unc_path, access_mode, create_options, out handle)
status, handle
status, handle

Figure 27: Sequence diagram detail for Open File DFS

1. The SMB Application invokes open_dfs() with the specified unc_path to direct the SMB File Client to gain access to the file and return a handle that the SMB Application can use in subsequent operations against the file. open_dfs() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server, using mechanisms appropriate to paths which may encounter DFS namespaces.

2. The SMB File Client uses the mechanisms of [MS-DFSC] 3.1.4.1 to determine if the path references a DFS namespace.

3. If the path does not reference a DFS namespace, operation continues at step 5.

4. The SMB File Client makes use of the mechanisms of [MS-DFSC] 3.1.4.1 to perform the SMB Application's request. These mechanisms include sequences equivalent to the Open File SMB sequence diagram of this document within a comprehensive state machine for DFS namespaces.

5. The path does not reference a DFS namespace.

6. The SMB File Client makes direct use of the Open File SMB sequence diagram, providing parameters as specified by the caller, returning the results to the SMB Application.

6.1.5.11.3 Final System State

- The SMB Application has a handle it can use when making subsequent requests to the SMB File Client for operation to be performed on the file.
- State described in the "Final System State" section of the sequence diagrams referenced in this sequence.

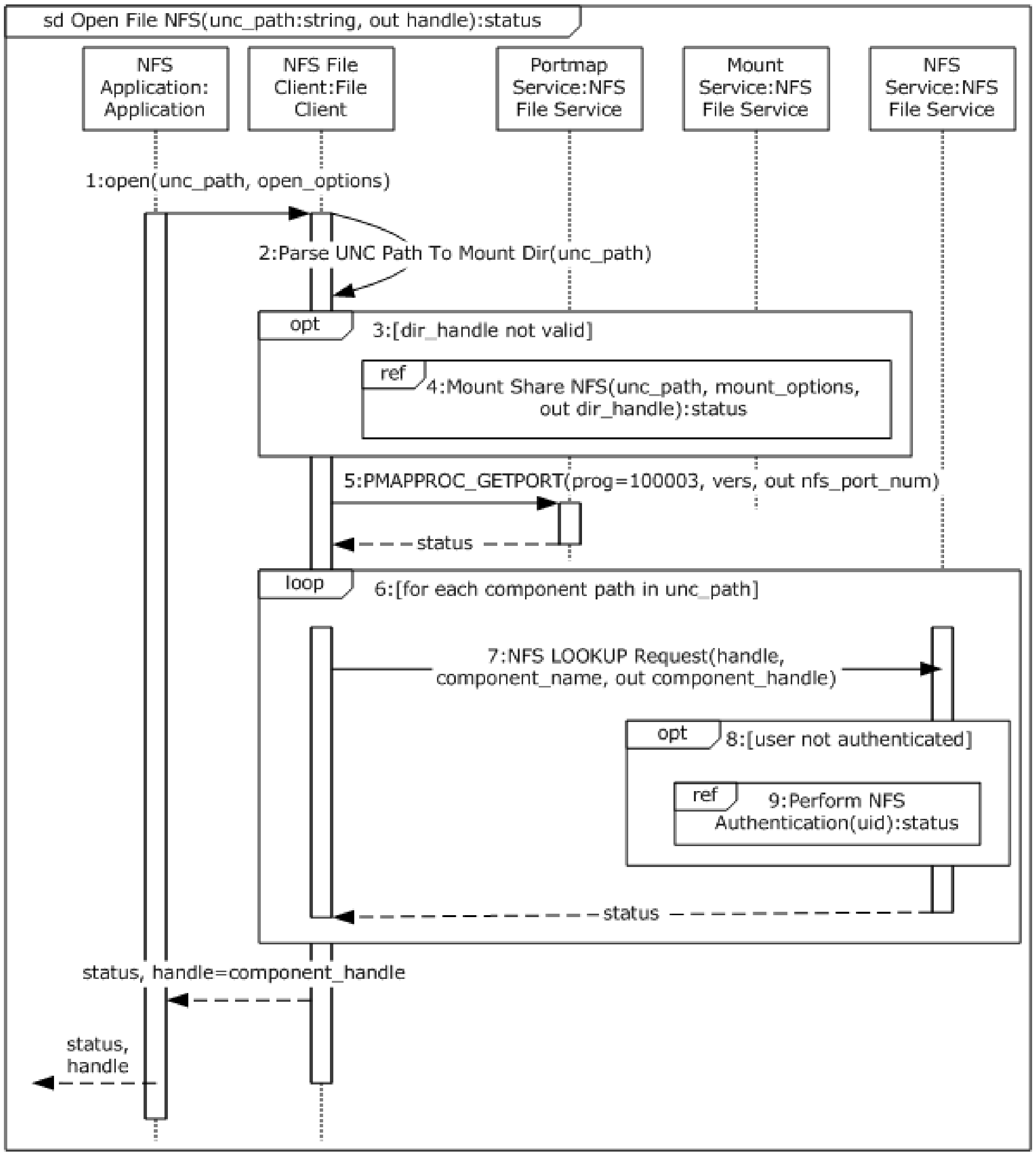
6.1.5.12 Open File NFS

The sequence described in this example details how the NFS Application uses the NFS File Client to open a file on the NFS File Share.

6.1.5.12.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.12.2 Sequence of Events



sd Open File NFS(unc_path:string, out handle):status
NFS Application: Application
NFS File Client:File Client
Portmap Service:NFS File Service
Mount Service:NFS File Service
NFS Service:NFS File Service
1:open(unc_path, open_options)
2:Parse UNC Path To Mount Dir(unc_path)
opt
3:[dir_handle not valid]
ref
4:Mount Share NFS(unc_path, mount_options, out dir_handle):status
5:PMAPPROC_GETPORT(prog=100003, vers, out nfs_port_num)
status
loop
6:[for each component path in unc_path]
7:NFS LOOKUP Request(handle, component_name, out component_handle)
opt
8:[user not authenticated]
ref
9:Perform NFS Authentication(uid):status
status
status, handle=component_handle
status, handle

Figure 28: Sequence diagram detail for Open File NFS

1. The NFS Application initiates open() on an UNC path to a file on an NFS File Share on the file server. open() is an example of an application specific message that the NFS Application initiates to direct the NFS File Client to communicate with the NFS File Service.

2. The NFS File Client extracts the file server name, NFS File Share and the relative path to the file to be opened from the unc_path as indicated in the message "Parse UNC Path To Mount Dir()".

3. The NFS File Client checks some internal state to determine whether it already has a handle to the NFS File Share as a result of a prior mount of the NFS File Share from the file server. If the

NFS File Client does not have an NFS handle corresponding to the NFS File Share as indicated by the guard condition (dir_handle not present), it proceeds to step 4.

4. The NFS File Client performs a mount of the NFS File Share from the file server, as described in the referenced sequence Mount Share NFS (section 6.1.5.10).

5. The NFS File Client contacts the Portmap Service on the file server to obtain the port number of the NFS File Service by sending the message PMAPPROC_GETPORT request as described in [RFC1833], section 3.2.

6. The NFS File Client then traverses the path to the file on the NFS File Share to obtain an NFS File Handle corresponding to the file.

7. The NFS File Client uses LOOKUP request as described in [RFC1813] section 3.3.3 to obtain NFS File Handle, component_handle, corresponding to the component path to the target file on the NFS File Share.

8. The NFS File Service checks its internal state to determine whether the user identity has been authenticated previously and whether it is still valid for the current message call. If the user identity has not been authenticated previously, as indicated by the guard condition "user not authenticated", it proceeds to step 9. Implementations of the NFS File Service can choose to cache authentication information for short durations instead of authenticating user identity on every message call.

9. The NFS File Service authenticates the user identity as described in the referenced sequence Perform NFS Authentication, section 6.1.5.23.

6.1.5.12.3 Final System State

- The NFS File Client successfully obtains the NFS File Handle corresponding to the file to be opened.

- The NFS Application successfully obtains a handle to the opened file.

6.1.5.13 Open File SMB

The sequence described in this example details how the SMB Application uses the SMB File Client to open a file on the SMB File Share.

6.1.5.13.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.

- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.13.2 Sequence of Events

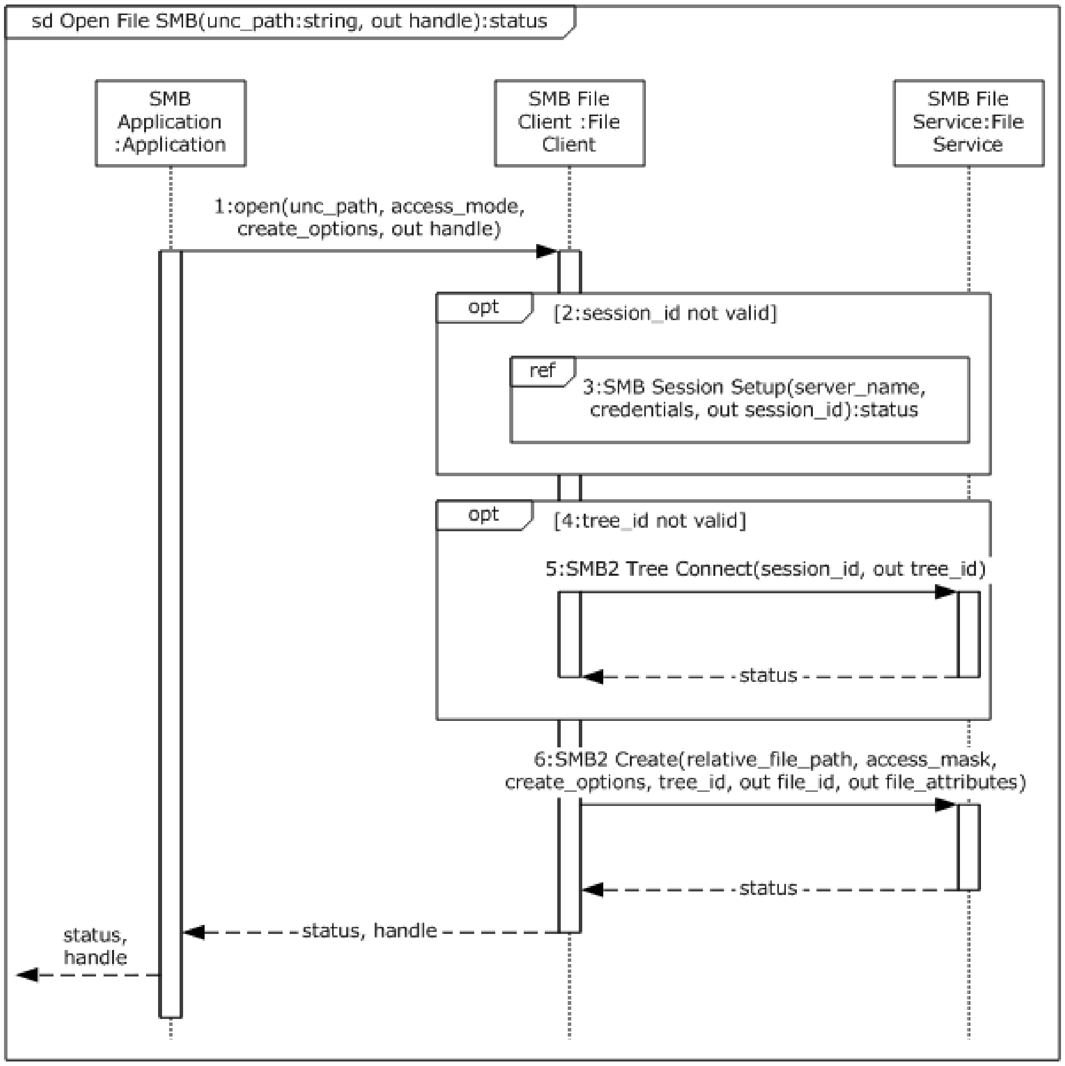

sd Open File SMB(unc_path:string, out handle):status
SMB Application :Application
SMB File Client :File Client
SMB File Service:File Service
1:open(unc_path, access_mode, create_options, out handle)
opt
[2:session_id not valid]
ref
3:SMB Session Setup(server_name, credentials, out session_id):status
opt
[4:tree_id not valid]
5:SMB2 Tree Connect(session_id, out tree_id)
status
6:SMB2 Create(relative_file_path, access_mask, create_options, tree_id, out file_id, out file_attributes)
status
status, handle
status, handle

Figure 29: Sequence diagram detail for Open File SMB

1. The SMB Application extracts the remote file server, the SMB File Share, and the target directory from the UNC path, unc_path, and invokes open() to direct the SMB File Client to gain access to the file and return a handle that the SMB Application can use in subsequent operations against the file. open() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server.

2. The SMB File Client examines internal private state to determine whether or not there is an existing session between the SMB File Client and the SMB File Service that can be use to satisfy the request from the SMB Application. If a session is required the SMB File Client proceeds to operation 3, otherwise it continues at operation 4.

3. The SMB File Client attempts to establish a session using the sequence SMB Session Setup described in section 6.1.5.24.

4. The SMB File Client examines internal private state to determine whether or not there is a valid tree connection for the SMB File Share between the SMB File Client and the SMB File Service. If there is no tree connection the SMB File Client proceeds to step 5, otherwise it continues to step 6. The process is further described in [MS-SMB2] section 3.2.4.2.4.

5. The SMB File Client attempts to establish a tree connection between the SMB File Client and the SMB File Service using an SMB2 TREE_CONNECT Request (see [MS-SMB2] section 2.2.9) using the previously obtained session identifier session_id and the UNC pathto the share. The SMB File Service sends an SMB2 TREE_CONECT Response (see [MS-SMB2] section 2.2.10) with details of the tree connection, including an identifier tree_id the SMB File Client can use to identify subsequent requests using the tree connection.

6. Once the SMB File Client has identified a suitable tree connection for the SMB File Share, it sends an SMB2 CREATE Request to the SMB File Service to request that a file be created. Once the SMB File Service has processed the SMB2 CREATE Request, it sends an SMB2 CREATE Response (see [MS-SMB2] section 2.2.14) which contains the result of the SMB2 CREATE Request. This process is described further in [MS-SMB2] section 3.2.4.3. If the operation was successful, the SMB File Client generates a handle which is returned to the SMB Application for use when performing subsequent operations against the file.

6.1.5.13.3 Final System State

- The SMB Application has a handle it can use when making subsequent requests to the SMB File Client for operation to be performed on the file.
- The SMB File Client has state established on the SMB File Service represented by the session_id, tree_id, and file_id data types returned.

6.1.5.14 Perform File Operation NFS Read

The sequence described in this example details how the NFS Application uses the NFS File Client to read the contents of a file on the NFS File Share.

6.1.5.14.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.14.2 Sequence of Events

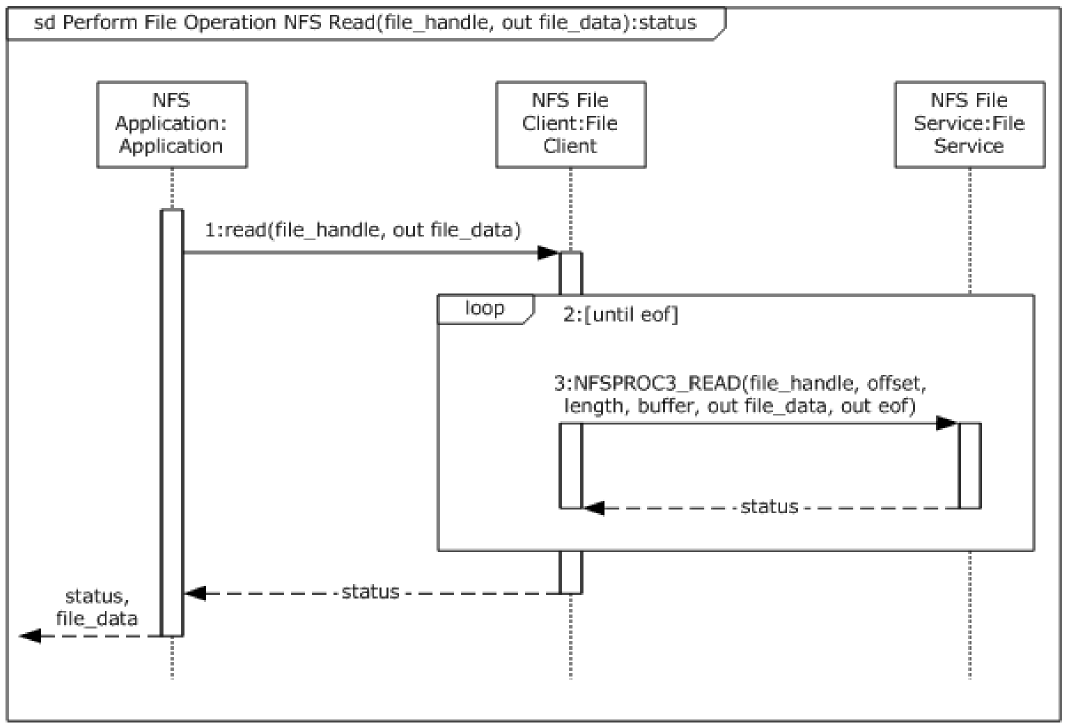

sd Perform File Operation NFS Read(file_handle, out file_data):status
NFS Application: Application
NFS File Client:File Client
NFS File Service:File Service
1:read(file_handle, out file_data)
loop
2:[until eof]
3:NFSPROC3_READ(file_handle, offset, length, buffer, out file_data, out eof)
status
status
status, file_data

Figure 30: Sequence diagram detail for Perform File Operation NFS Read

1. The NFS Application makes a read file request to the NFS File Client using read() with a file handle and a buffer to receive the data. read() is an example of an application specific message that the NFS Application initiates to direct the NFS File Client to communicate with the NFS File Service on a file server.

2. The NFS File Client gets the complete file contents by issuing a series of NFS requests to the NFS File Service.

3. The NFS File Client uses the NFSPROC3_READ message as described in [RFC1813], section 3.3.6 to read the file data.

4. The NFS File Service checks its internal state to determine whether the user identity has been authenticated previously and whether it is still valid for the current message call. This is indicated by the guard condition "user not authenticated". Implementations of the NFS File Service can choose to cache authentication information for short durations instead of authenticating user identity on every message call.

5. The NFS File Service authenticates the user identity as described in the referenced sequence Perform NFS Authentication.

6.1.5.14.3 Final System State

The NFS Application successfully obtains the file contents from the NFS File Share on the file server.

6.1.5.15 Perform File Operation NFS Write

The sequence described in this example details how the NFS Application uses the NFS File Client to update the contents of a file on the NFS File Share.

6.1.5.15.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.15.2 Sequence of Events

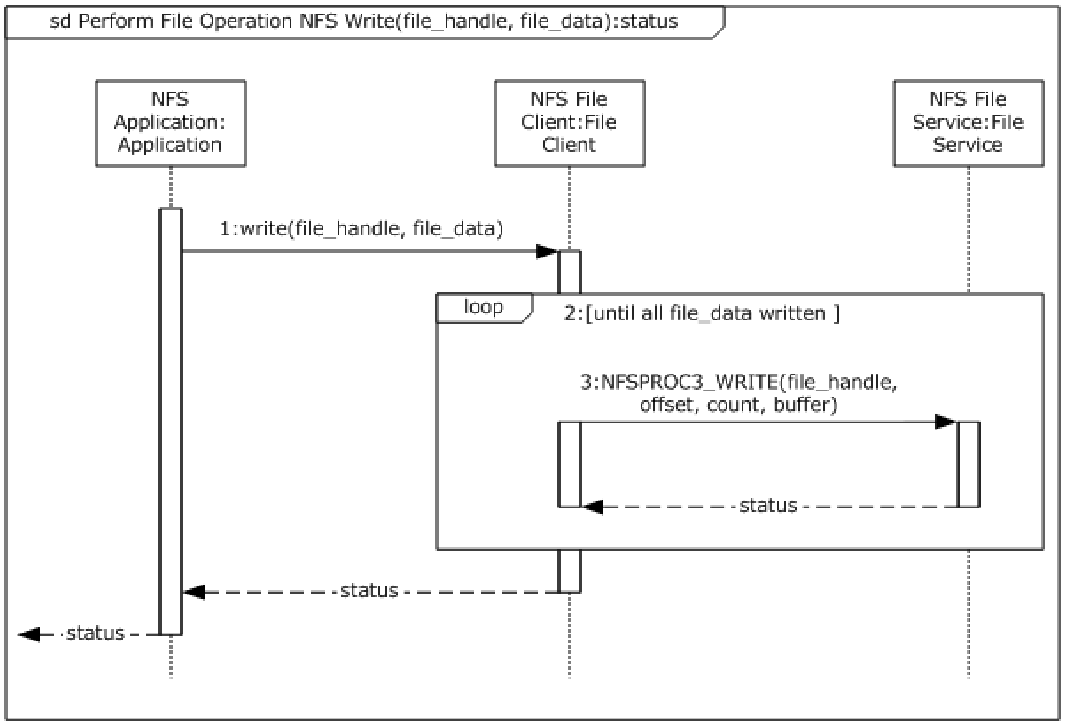

sd Perform File Operation NFS Write(file_handle, file_data):status
NFS Application: Application
NFS File Client:File Client
NFS File Service:File Service
1:write(file_handle, file_data)
loop
2:[until all file_data written]
3:NFSPROC3_WRITE(file_handle, offset, count, buffer)
status
status
status

Figure 31: Sequence diagram detail for Perform File Operation NFS Write

1. The NFS Application makes a read file request to the NFS File Client using read() with a file handle and a buffer to receive the data. read() is an example of an application specific message that the NFS Application initiates to direct the NFS File Client to communicate with the NFS File Service on a file server.

2. The NFS File Client writes the requested file contents, as represented by file_data, by issuing a series of NFS requests to the NFS File Service.

3. The NFS File Client uses the NFSPROC3_WRITE message as described in [RFC1813], section 3.3.7 to write the provided file_data.

4. The NFS File Service checks its internal state to determine whether the user identity has been authenticated previously and whether it is still valid for the current message call. This is indicated by the guard condition "user not authenticated". Implementations of the NFS File Service can

choose to cache authentication information for short durations instead of authenticating user identity on every message call.

5. The NFS File Service authenticates the user identity as described in the referenced sequence Perform NFS Authentication.

6.1.5.15.3 Final System State

The NFS Application successfully updates the file contents on the NFS File Share at the file server.

6.1.5.16 Perform File Operation SMB Close

The sequence described in this example details how the SMB Application uses the SMB File Client to close a previously open handle to a file or directory on the SMB File Share.

6.1.5.16.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.16.2 Sequence of Events

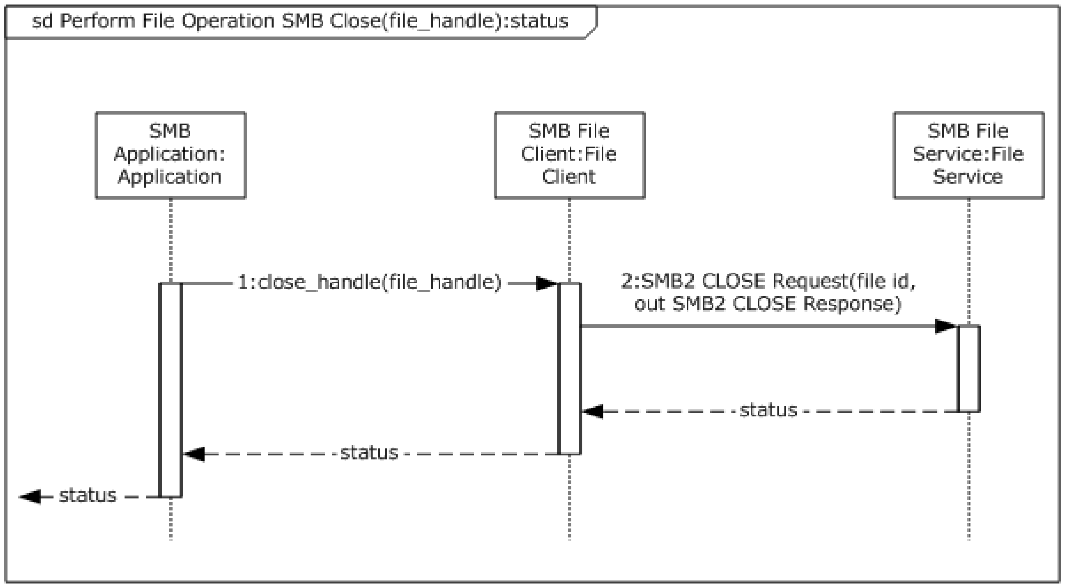

sd Perform File Operation SMB Close(file_handle):status
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
1:close_handle(file_handle)
2:SMB2 CLOSE Request(file id, out SMB2 CLOSE Response)
status
status
status

Figure 32: Sequence diagram detail for Perform File Operation SMB Close

1. The SMB Application makes a close request using close_handle() passing a file handle that was previously issued by the SMB File Client. close_handle() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server.

2. The SMB File Client uses internally held state and the supplied file handle to obtain the file_id used to identify the file to the SMB File Service. The SMB File Client sends an SMB2 CLOSE Request (see [MS-SMB2] section 2.2.15) to the SMB File Service to notify the SMB File Service that no further access is required to the file identified by file_id and that any resources held by

the SMB File Service to process requests for the file on behalf of the SMB File Client may be released. The SMB File Service sends an SMB2 CLOSE Response message back to the SMB File Client to indicate the SMB File Service has processed the SMB2 CLOSE Request. This process is further described in [MS-SMB2] section 3.3.5.10.

6.1.5.16.3 Final System State

- The SMB File Client and SMB File Service update their internal state corresponding to the closed file handle.
- The file identified by the file handle is closed and the SMB Application may make no further use of the file handle.
- The file identified to the SMB File Client using file_id is closed and the SMB File Client may make no further use of file_id.

6.1.5.17 Perform File Operation SMB Enumerate Directory

The sequence described in this example details how the SMB Application uses the SMB File Client to enumerate a directory through a previously opened handle on the SMB File Share.

6.1.5.17.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.17.2 Sequence of Events

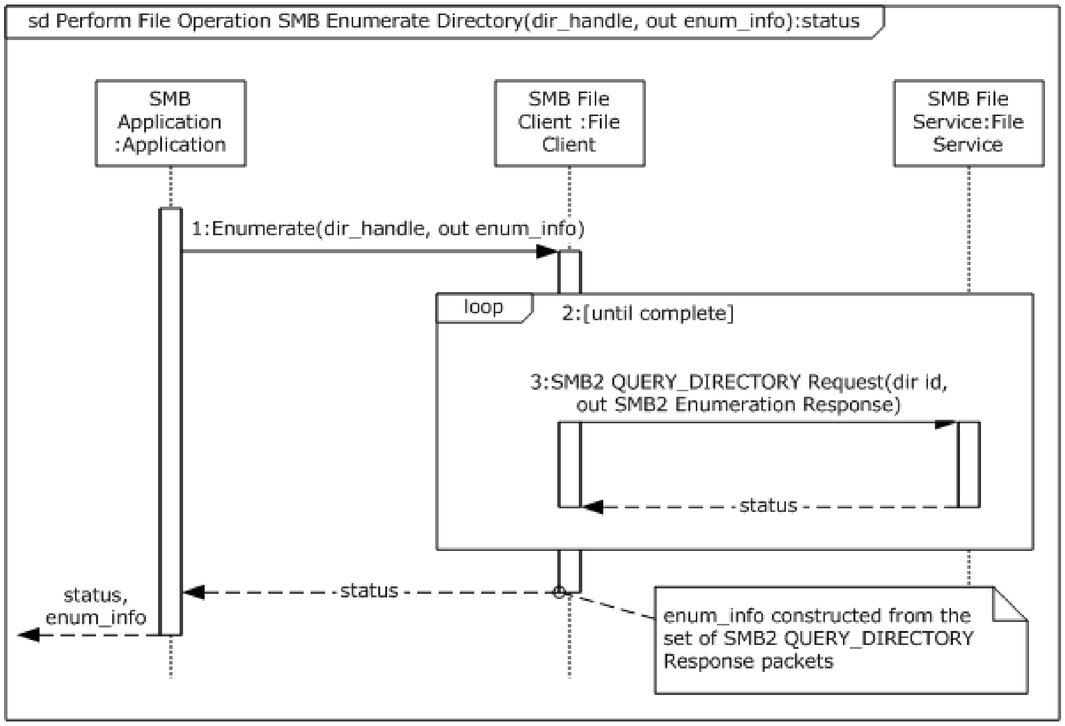

sd Perform File Operation SMB Enumerate Directory(dir_handle, out enum_info):status
SMB Application :Application
SMB File Client :File Client
SMB File Service:File Service
1:Enumerate(dir_handle, out enum_info)
loop
2:[until complete]
3:SMB2 QUERY_DIRECTORY Request(dir id, out SMB2 Enumeration Response)
status
status
status, enum_info
enum_info constructed from the set of SMB2 QUERY_DIRECTORY Response packets

Figure 33: Sequence diagram detail for Perform File Operation SMB Enumerate Directory

1. The SMB Application makes an enumerate request using enumerate(), passing a directory handle that was previously issued by the SMB File Client along with a buffer to receive enumeration results. enumerate() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server.

2. The SMB File Client repeats step 3 until the enumeration is complete. This is determined by the SMB File Service returning a status code other than STATUS_SUCCESS. A status of STATUS_NO_MORE_ITEMS indicates successful completion of the enumeration, while any other status indicates failure.

3. The SMB File Client uses internally held state and the supplied directory handle to obtain the dir_id used to identify the directory to the SMB File Service. The SMB File Client sends an SMB2 QUERY_DIRECTORY Request (see [MS-SMB2] section 2.2.33) to the SMB File Service describing the requested enumeration. On the initial step of this loop the client specifies the SMB2_REOPEN flag in the request, indicating that the server should initiate a new enumeration, discarding any previous enumeration state. The SMB File Service sends an SMB2 QUERY_DIRECTORY Response message back to the SMB File Client containing the requested enumeration results. This process is further described in [MS-SMB2] section 3.3.5.18.

6.1.5.17.3 Final System State

The SMB Application successfully obtains the directory contents from the SMB File Share on the file server.

6.1.5.18 Perform File Operation SMB Query FS Information

The sequence described in this example details how the SMB Application uses the SMB File Client to query file system attributes on the SMB File Share.

6.1.5.18.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.18.2 Sequence of Events

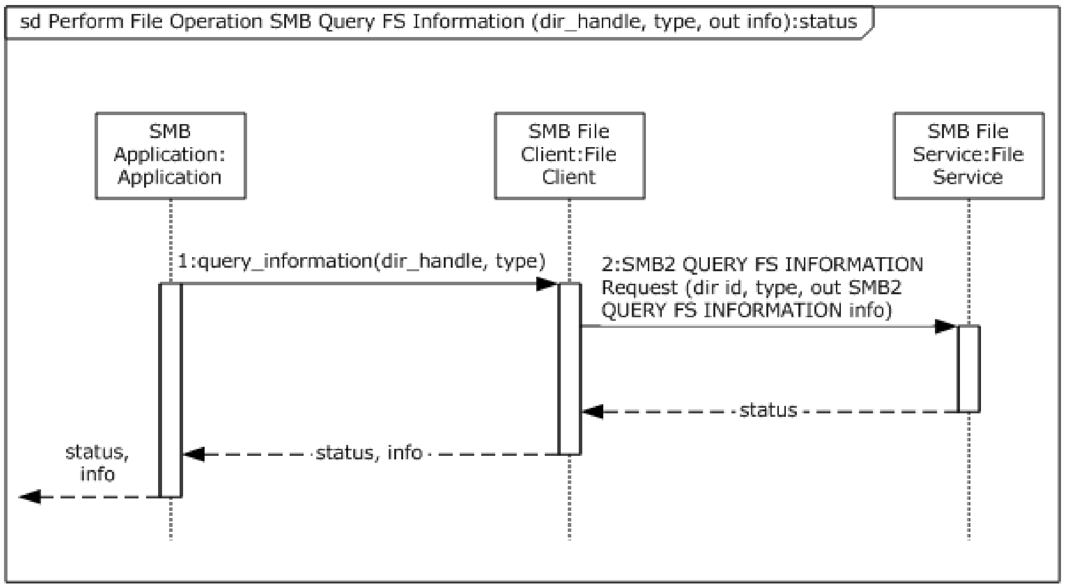

sd Perform File Operation SMB Query FS Information (dir_handle, type, out info):status
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
1:query_information(dir_handle, type)
2:SMB2 QUERY FS INFORMATION Request (dir id, type, out SMB2 QUERY FS INFORMATION info)
status
status, info
status, info

Figure 34: Sequence diagram detail for Perform File Operation SMB Query FS Information

1. The SMB Application makes a read file request to the SMB File Client using query_information() and a file handle that was previously issued to the SMB Application by the SMB File Client for the file of interest. query_information() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server.
2. The SMB File Client uses internally held state and the supplied file handle to obtain the file_id used to identify the file to the SMB File Service. The SMB File Client sends an SMB2 QUERY_INFO Request (see [MS-SMB2] section 2.2.37), with InfoType set to SMB2_0_INFO_FILESYSTEM, to the SMB File Service.

6.1.5.18.3 Final System State

On successful completion of the SMB2 QUERY_INFO Request, the SMB Application returns the information to the invoker of the sequence.

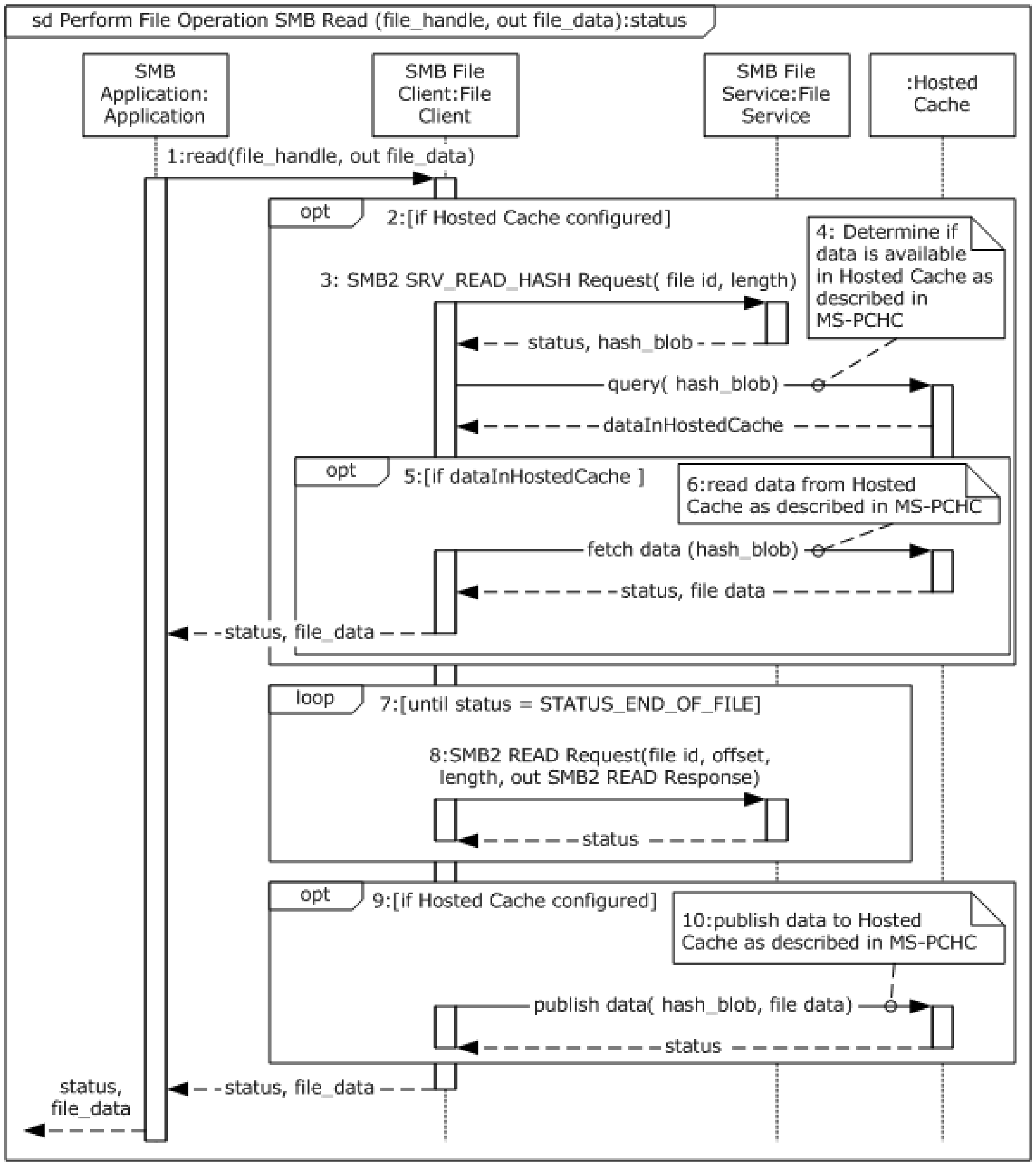
6.1.5.19 Perform File Operation SMB Read

The sequence described in this example details how the SMB Application uses the SMB File Client to read the contents of a file on the SMB File Share.

6.1.5.19.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.19.2 Sequence of Events



sd Perform File Operation SMB Read (file_handle, out file_data):status
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
:Hosted Cache
1:read(file_handle, out file_data)
opt
2:[if Hosted Cache configured]
3: SMB2 SRV_READ_HASH Request(file id, length)
4: Determine if data is available in Hosted Cache as described in MS-PCHC
status, hash_blob
query(hash_blob)
dataInHostedCache
opt
5:[if dataInHostedCache]
6:read data from Hosted Cache as described in MS-PCHC
fetch data (hash_blob)
status, file data
status, file_data
loop
7:[until status = STATUS_END_OF_FILE]
8:SMB2 READ Request(file id, offset, length, out SMB2 READ Response)
status
opt
9:[if Hosted Cache configured]
10:publish data to Hosted Cache as described in MS-PCHC
publish data(hash_blob, file data)
status
status, file_data
status, file_data

Figure 35: Sequence diagram detail for Perform File Operation SMB Read

1. The SMB Application makes a read file request to the SMB File Client using read() with a file handle and a buffer to receive the data. read() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server.

2. If the SMB File Client is configured to use Hosted Cache and the share supports hash generation (see [MS-SMB2] 2.2.10) then the SMB File Client may attempt to source the requested data from the Hosted Cache. If not, then operation continues at step 7.

3. The SMB File Client requests the hash blob for the file data to be read from the SMB File Service.

4. The SMB File Client communicates with the Hosted Cache, supplying the hash blob, to determine if the requested data is currently available in the Hosted Cache. See [MS-PCHC] for a description of how to interface with the Hosted Cache.

5. If the required data is not available in the Hosted Cache, operation continues at step 7.

6. The SMB File Client retrieves the required data from the Hosted Cache. See [MS-PCHC] for a description of how to interface with the Hosted Cache. The data is then returned to SMB Application and operation is complete at this point.

7. The SMB File Client retrieves the requested data from the SMB File Service. The file to be read may be larger than can be processed by a single SMB2 READ Request/SMB2 READ Response pair so the SMB File Client should be prepared to issue an iteration of SMB2 READ Requests until the complete file has been retrieved.

8. The SMB File Client uses internally held state and the supplied file handle to obtain the file_id used to identify the file to the SMB File Service. The SMB File Client sends an SMB2 READ Request (see [MS-SMB2] section 2.2.19) to the SMB File Service for a piece of the file. A single piece of the file may be defined by the offset of the start of the piece from the beginning of the file and the length of the piece to be retrieved. This is further described in [MS-SMB2] section 3.2.4.5. The size of the piece of the file to be retrieved is limited by the upper limit of a single SMB2 READ Response (see [MS-SMB2] section 2.2.20) as supplied by the SMB File Service to the SMB File Client in the SMB2 NEGOTIATE Response (see [MS-SMB2] section 2.2.4) during the session establishment negotiation sequence.

9. If the SMB File Client is configured to use the Hosted Cache and the share supports hash generation (see [MS-SMB2] 2.2.10), then the SMB File Client may attempt to publish the data retrieved in steps 7 and 8 into the Hosted Cache, for the benefit of other SMB File Clients. If not, the data is returned to SMB Application, and operation is complete.

10. The SMB File Client publishes the data to the Hosted Cache. See [MS-PCHC] for a description of how to interface with the Hosted Cache.

6.1.5.19.3 Final System State

- If the SMB File Client is configured to use Hosted Cache, the file share supports hash generation, and the data was not present in Hosted Cache but was successfully read from the SMB File Service, then the SMB File Client may have published the data into the Hosted Cache.

- On successful retrieval of the data from the SMB File Client, from either the SMB File Server or Hosted Cache, the SMB Application returns a copy of the file_data to the invoker of the sequence.

6.1.5.20 Perform File Operation SMB Request Directory Change Notification

The sequence described in this example details how the SMB Application uses the SMB File Client to request directory change notifications from the SMB File Service in a directory on the SMB File Share.

6.1.5.20.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.

- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.20.2 Sequence of Events

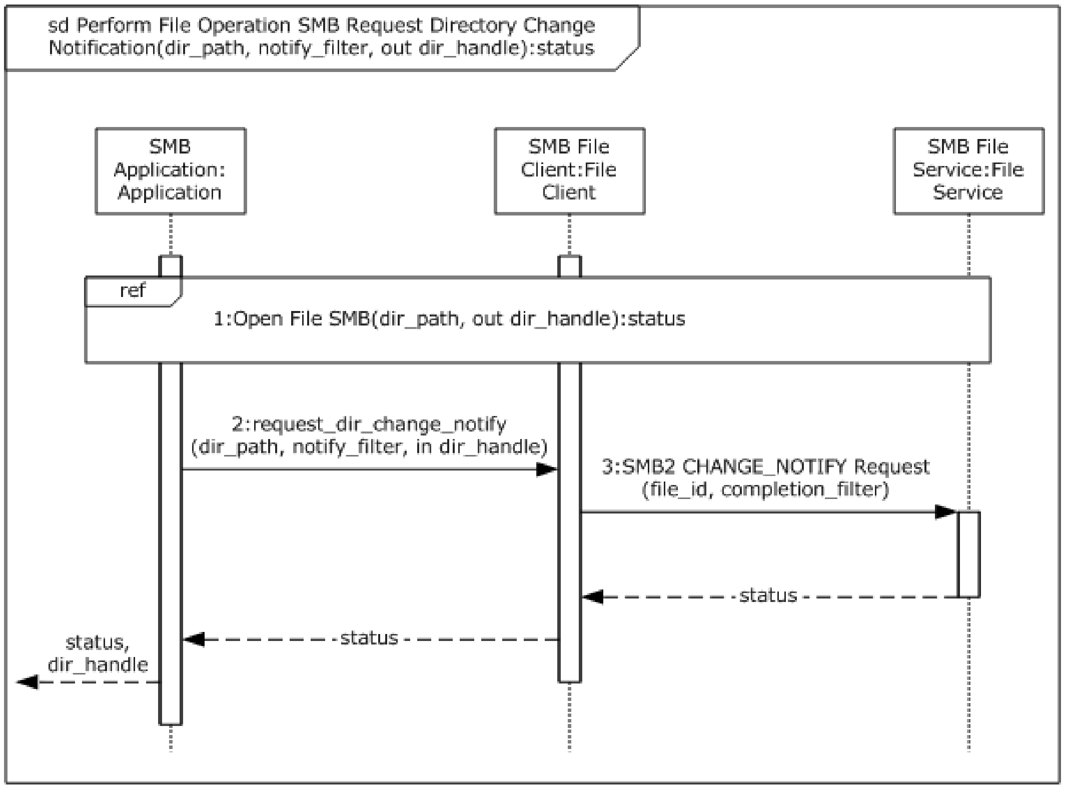

sd Perform File Operation SMB Request Directory Change Notification(dir_path, notify_filter, out dir_handle):status
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
ref
1:Open File SMB(dir_path, out dir_handle):status
2:request_dir_change_notify (dir_path, notify_filter, in dir_handle)
3:SMB2 CHANGE_NOTIFY Request (file_id, completion_filter)
status
status
status, dir_handle

Figure 36: Sequence diagram detail for Perform File Operation SMB Request Directory Change Notification

1. The SMB File Client opens a handle to the directory path on the SMB File Service as covered in the sequence Open File SMB.

2. The SMB Application registers for a change notification using request_dir_change_notify() passing the handle to the directory and a notification filter, notify_filter, that contains the type of notifications it is supposed to receive. request_dir_change_notify() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to register for a directory change notification on the SMB File Service on a file server.

3. The SMB Client sends a SMB2 CHANGE NOTIFY Request (see [MS-SMB2] section 2.2.35) to the SMB File Service to register a change notification on the directory that was successfully opened in the previous step. On successful completion of the SMB2 CHANGE NOTIFY Request the SMB File Client returns the directory handle, dir_handle, back to the SMB Application. The SMB Application may use dir_handle to wait for notifications indicating changes in the requested state. This is described further in [MS-SMB2] section 3.3.5.19.

6.1.5.20.3 Final System State

- The SMB Application receives a handle to the directory which it can wait to receive notifications on.

- The SMB File Service successfully registers for change notification on the directory with the underlying Object Store.

6.1.5.21 Perform File Operation SMB Write

The sequence described in this example details how the SMB Application uses the SMB File Client to update the contents of a file on the SMB File Share.

6.1.5.21.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.21.2 Sequence of Events

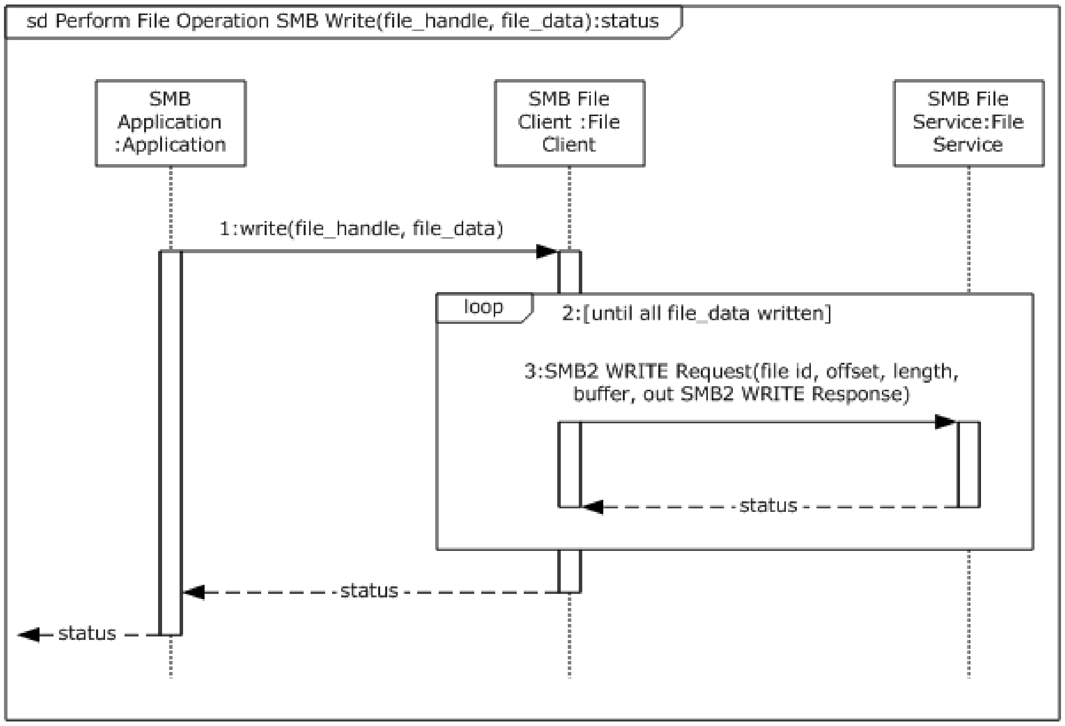

sd Perform File Operation SMB Write(file_handle, file_data):status
SMB Application :Application
SMB File Client :File Client
SMB File Service:File Service
1:write(file_handle, file_data)
loop
2:[until all file_data written]
3:SMB2 WRITE Request(file id, offset, length, buffer, out SMB2 WRITE Response)
status
status
status

Figure 37: Sequence diagram detail for Perform File Operation SMB Write

1. The SMB Application makes a write file request to the SMB File Client using write() with a file handle and the data to be written to the file. write() is an example of an application specific message that the SMB Application initiates to direct the SMB File Client to communicate with the SMB File Service on a file server.
2. The file to be written may be larger than can be processed by a single SMB2 WRITE Request/SMB2 WRITE Response pair so the SMB File Client should be prepared to issue an iteration of SMB2 WRITE Requests until the complete file has been written.
3. The SMB File Client uses internally held state and the supplied file handle to obtain the file_id used to identify the file to the SMB File Service. The SMB File Client sends an SMB2 WRITE Request (see [MS-SMB2] section 2.2.21) to the SMB File Service for a piece of the file. A single piece of the file may be defined by the offset of the start of the piece from the beginning of the file and the length of the piece to be written. This is further described in [MS-SMB2] section

3.2.4.6. The size of the piece of the file to be retrieved is limited by the upper limit of a single SMB2 WRITE Request as supplied by the SMB File Service to the SMB File Client in the SMB2 NEGOTIATE Response (see [MS-SMB2] section 2.2.4) during the session establishment negotiation sequence.

6.1.5.21.3 Final System State

On successful completion of the set of SMB2 WRITE Request, SMM2 WRITE Response pairs, the file_data supplied by the SMB Application has been transferred by the SMB File Client to the SMB File Service and stored in the object store backing the SMB File Service.

6.1.5.22 Perform File Operation SMB Delete

The sequence described in this example details how the SMB Application uses the SMB File Client to delete a file on the SMB File Share.

6.1.5.22.1 Initial System State

The general requirements are described in System Assumptions and Preconditions (section 4.2) and in the "Initial System State" section of the sequence diagram in section 6.1.5.22.2.

6.1.5.22.2 Sequence of Events

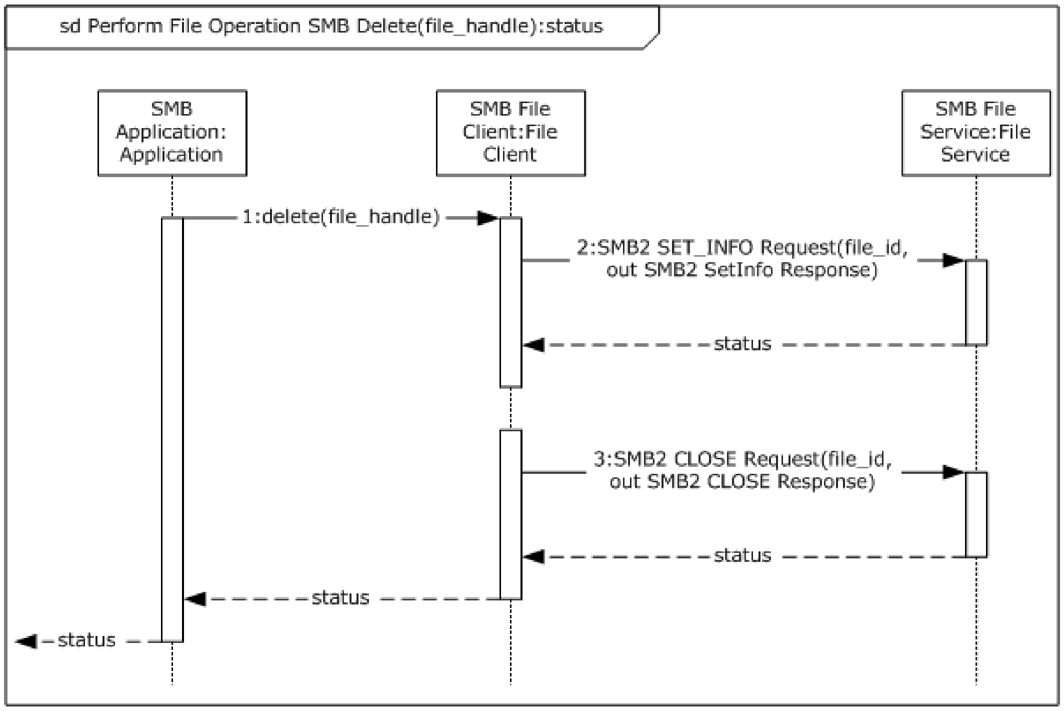

sd Perform File Operation SMB Delete(file_handle):status
SMB Application: Application
SMB File Client:File Client
SMB File Service:File Service
1:delete(file_handle)
2:SMB2 SET_INFO Request(file_id, out SMB2 SetInfo Response)
status
3:SMB2 CLOSE Request(file_id, out SMB2 CLOSE Response)
status
status
status

Figure 38: Sequence Diagram Detail for Perform File Operation SMB Delete

The sequence of events to perform an SMB delete operation is as follows:

1. The SMB application makes a delete request using delete(), passing a file handle that was previously issued by the SMB File Client. The delete() message is an example of an application-

specific message that the SMB application initiates to direct the SMB File Client to communicate with the **SMB File Service** on a file server.

2. The SMB File Client uses internally held state and the supplied file handle to obtain the file_id used to identify the file to the **SMB File Service**. The SMB File Client sends an SMB2 SetInfo Request (see [MS-SMB2] section 2.2.39) to the SMB File Service to notify the SMB File Service to mark the file identified by file_id for deletion. The SMB File Service sends an SMB2 SetInfo Response message back to the SMB File Client to indicate that the SMB File Service has processed the SMB2 SetInfo Request. This process is further described in [MS-SMB2] section 3.3.5.21.
3. The SMB File Clientt sends an SMB2 CLOSE Request (see [MS-SMB2] section 2.2.15) to the **SMB File Service** to notify the **SMB File Service** that no further access is required to the file identified by file_id and that any resources held by the **SMB File Service** to process requests for the file on behalf of the SMB File Client may be released. The **SMB File Service** sends an SMB2 CLOSE Response message back to the SMB File Client to indicate that the **SMB File Service** has processed the SMB2 CLOSE Request. This process is further described in [MS-SMB2] section 3.3.5.10.

6.1.5.22.3 Final System State

The final system state is defined by the following events:

- The file identified by file_id is deleted.
- The SMB File Client and **SMB File Service** close the file handle and update their internal state corresponding to it. The SMB application may make no further use of the file handle.
- The SMB File Client closes file_id and MAY make no further use of file_id.

6.1.5.23 Perform NFS Authentication

The sequence described in this example details how the NFS File Service authenticates a UID sent by the NFS Client.

6.1.5.23.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.

6.1.5.23.2 Sequence of Events

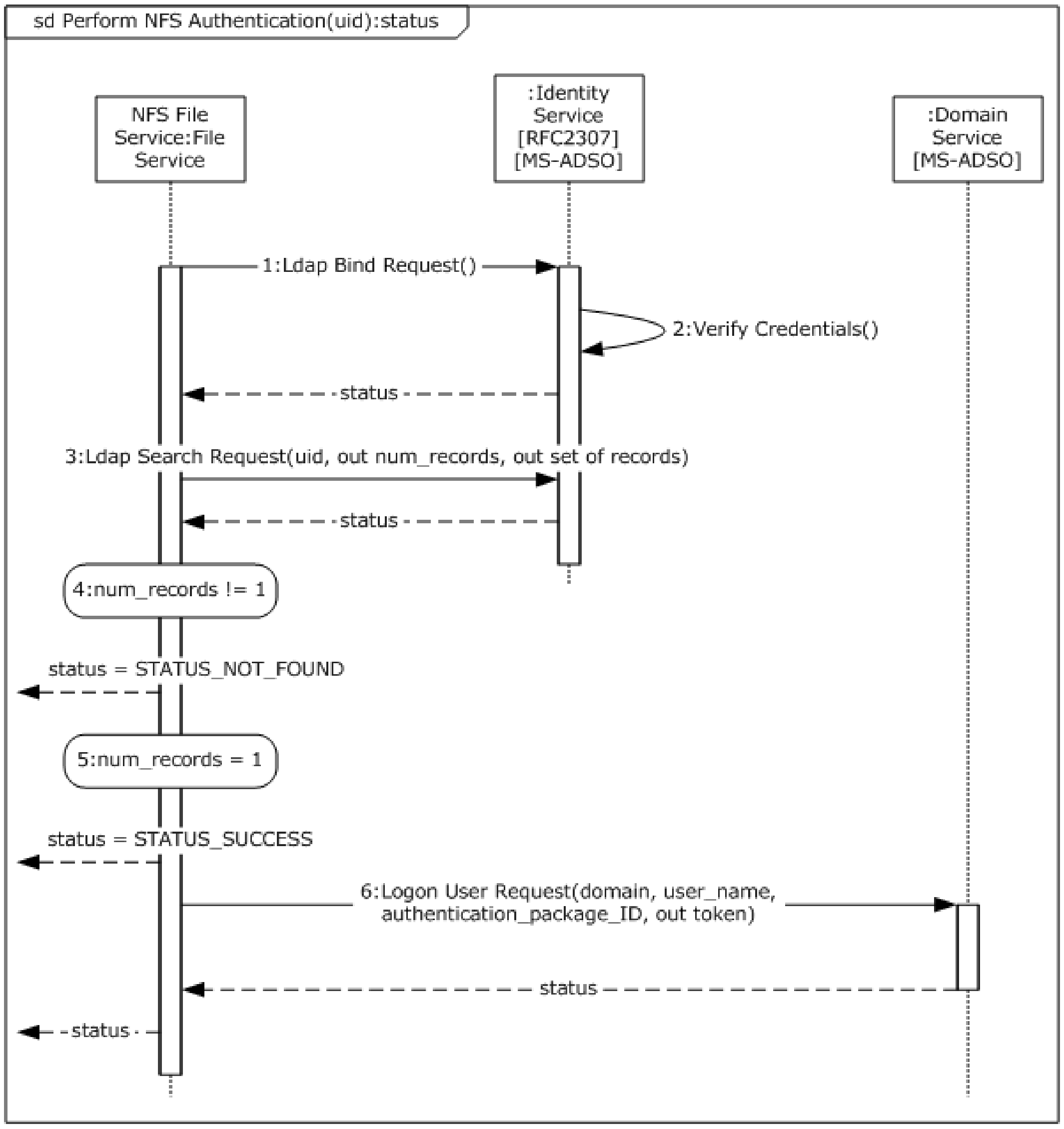

sd Perform NFS Authentication(uid):status
NFS File Service:File Service
:Identity Service [RFC2307] [MS-ADSO]
:Domain Service [MS-ADSO]
1:Ldap Bind Request()
2:Verify Credentials()
status
3:Ldap Search Request(uid, out num_records, out set of records)
status
4:num_records != 1
status = STATUS_NOT_FOUND
5:num_records = 1
status = STATUS_SUCCESS
6:Logon User Request(domain, user_name, authentication_package_ID, out token)
status
status

Figure 39: Sequence diagram detail for Perform NFS Authentication

1. The NFS File Service binds to the configured Identity Service using ldap_bind_request(), which is an example of an implementation specific message that is provided by the Identity Service such as described in [RFC2251] section 4.2.

2. The Identity Service verifies the credentials of the incoming bind request. verify_credentials() is an example of a implementation specific mechanism that is used by the Identity Service to authorize the caller of an ldap_bind_request().

3. The NFS File Service sends an ldap_search_request() message to the Identity Service to translate the supplied NFS identity (UID) to an account name that can be presented to the Authentication Service. The result of this message is a list of zero or more account names, set of records, and a count. num_records. ldap_search_request() is an example of an implementation

specific message that is provided by the Identity Service such as described in [RFC2251] section 4.5.

4. Whenever there is no unique translation from the supplied NFS identity, uid, as indicated by num_records set to 0x1, the NFS File Service fails the authentication and returns STATUS_NOT_FOUND back to the caller.
5. Whenever there is a unique translation from the supplied NFS identity, uid, as indicated by num_records set to 0x1, NFS File Service continues the authentication process.
6. The NFS File Service sends a logon_user_request() message to the Authentication Service to authenticate the user identified by the account name obtained as a result of the ldap_search_request(). logon_user_request() is an example of an implementation specific message flow as described in section 6 of [MS-AUTHSO]. The result of a successful completion of logon_user_request() is a security token, token, used in subsequent operations to identify the user account, user_name.

6.1.5.23.3 Final System State

The NFS File Service receives a token for the user account corresponding to the supplied NFS identity, uid.

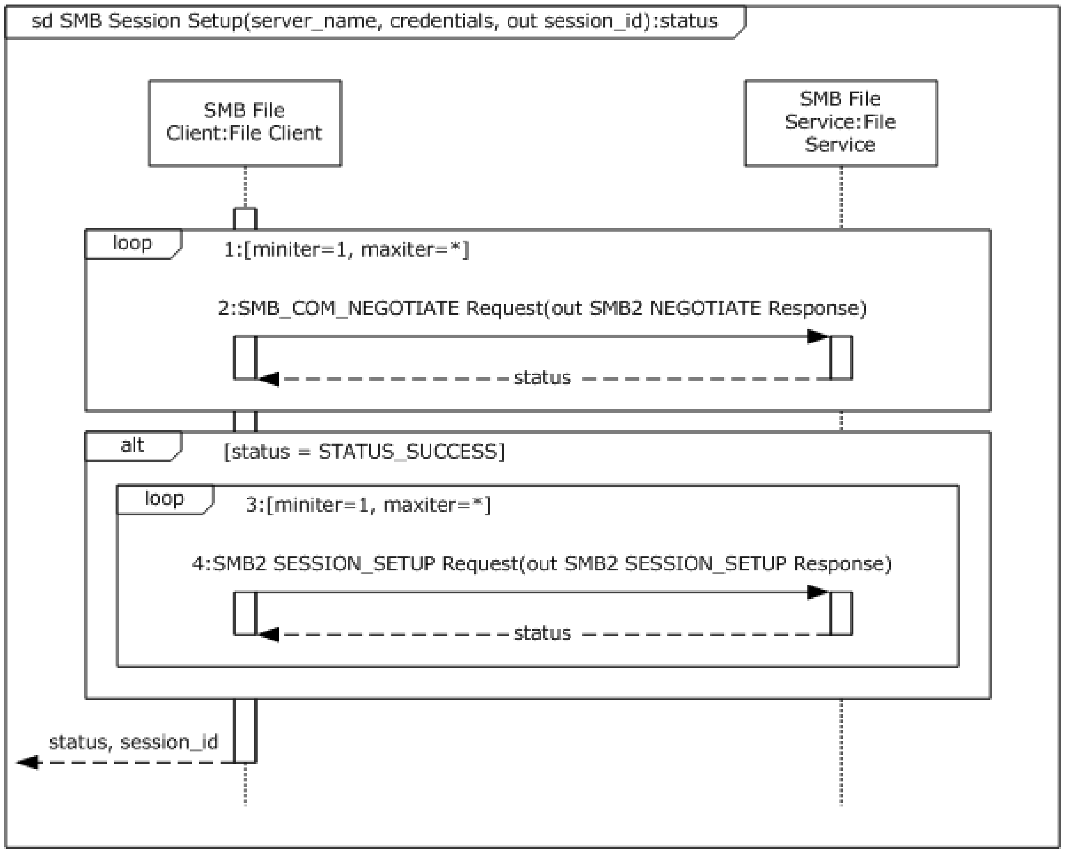
6.1.5.24 SMB Session Setup

The sequence described in this example details an SMB2 session establishment between the SMB File Client and the SMB File Service.

6.1.5.24.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.
- Requirements described in Task Assumptions and Preconditions in section 6.2.3 of [MS-AUTHSO].
- No SMB2 session exists between the SMB File Client and the SMB File Service, or the system has determined that a new session should be established between the SMB File Client and the SMB File Service.

6.1.5.24.2 Sequence of Events



sd SMB Session Setup(server_name, credentials, out session_id):status
SMB File Client:File Client
SMB File Service:File Service
loop
1:[miniter=1, maxiter=*]
2:SMB_COM_NEGOTIATE Request(out SMB2 NEGOTIATE Response)
status
alt
[status = STATUS_SUCCESS]
loop
3:[miniter=1, maxiter=*]
4:SMB2 SESSION_SETUP Request(out SMB2 SESSION_SETUP Response)
status
status, session_id

Figure 40: Sequence diagram detail for SMB Session Setup

1. The SMB File Client negotiates the SMB dialect and properties to be used for the connection by means of an iteration of one or more protocol messages.

2. One or more SMB_COM_NEGOTIATE Requests as described in [MS-CIFS] section 2.2.4.52 are sent from the SMB File Client to the SMB File Service and SMB2 NEGOTIATE Responses as described in [MS-SMB2] section 2.2.4 are sent from the SMB File Service to the SMB File Client. This is described further in [MS-SMB2] section 3.2.4.2.2. Once the negotiation has completed successfully, a connection exists between the SMB File Client and the SMB File Service.

3. On successful completion of the connection establishment, the SMB File Client attempts to establish a session between the SMB File Client and the SMB File Service by means of an iteration of one or more protocol messages.

4. One or more SMB2 SESSION_SETUP Requests as described in [MS-SMB2] section 2.2.5 are sent from the SMB File Client to the SMB File Service, resulting in SMB2 SESSION_SETUP Responses as described in [MS-SMB2] section 2.2.6 until the identity of the SMB User has been authenticated and the establishment of the session authorized against the SMB File Service is completed. This is described further in [MS-SMB2] section 3.2.4.2.3.

6.1.5.24.3 Final System State

- An authenticated and authorized session exists between the SMB File Client and the SMB File Service.
- There is an entry in the SMB File Server Connection List section 5.1.3.3.1 for the newly created connection.
- There is an entry in the SMB File Server Session List section 5.1.3.3.2 for the newly created session.

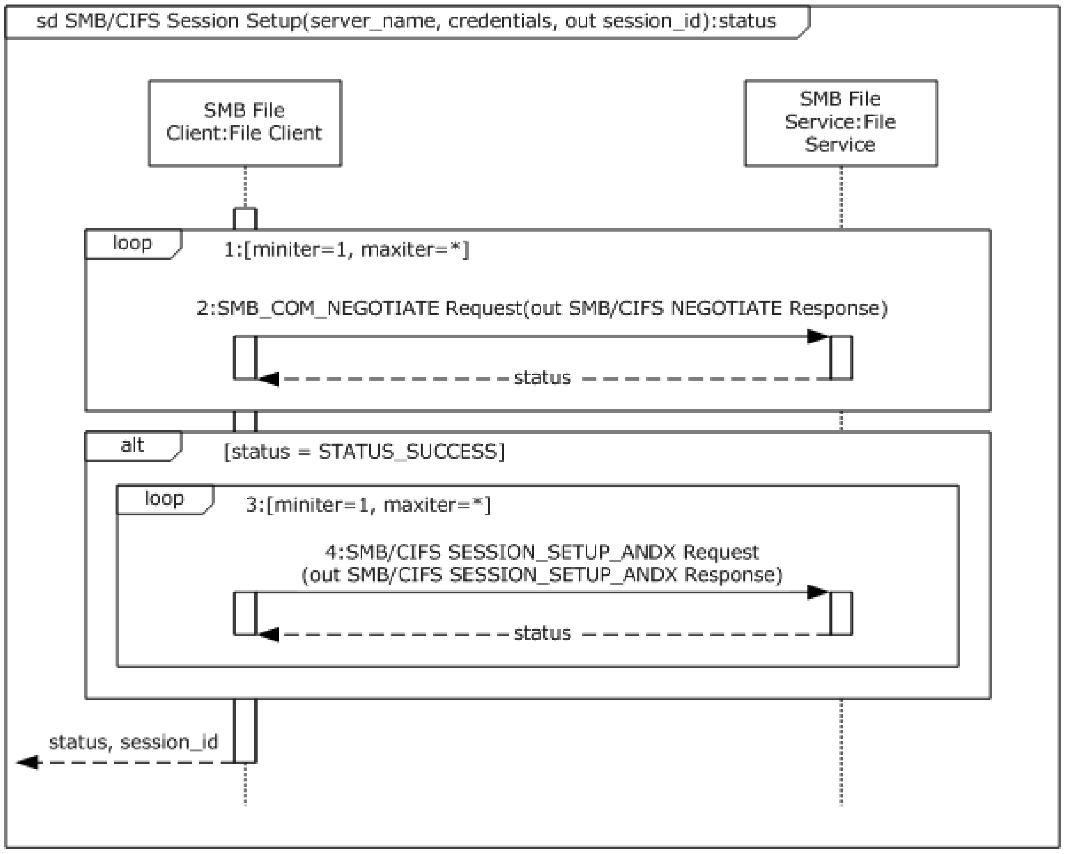
6.1.5.25 SMB/CIFS Session Setup

The sequence described in this example details an [MS-SMB] or [MS-CIFS] session establishment between the SMB File Client and the SMB File Service.

6.1.5.25.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.
- Requirements described in Task Assumptions and Preconditions in section 6.2.3 of [MS-AUTHSO].
- No [MS-SMB] or [MS-CIFS] session exists between the SMB File Client and the SMB File Service, or the system has determined that a new session should be established between the SMB File Client and the SMB File Service.

6.1.5.25.2 Sequence of Events



sd SMB/CIFS Session Setup(server_name, credentials, out session_id):status
SMB File Client:File Client
SMB File Service:File Service
loop
1:[miniter=1, maxiter=*]
2:SMB_COM_NEGOTIATE Request(out SMB/CIFS NEGOTIATE Response)
status
alt
[status = STATUS_SUCCESS]
loop
3:[miniter=1, maxiter=*]
4:SMB/CIFS SESSION_SETUP_ANDX Request (out SMB/CIFS SESSION_SETUP_ANDX Response)
status
status, session_id

Figure 41: Sequence diagram detail for SMB/CIFS Session Setup

1. The SMB File Client negotiates the SMB dialect and properties to be used for the connection by means of an iteration of one or more protocol messages.

2. One or more SMB_COM_NEGOTIATE Requests as described in [MS-CIFS] section 2.2.4.52 are sent from the SMB File Client to the SMB File Service and SMB NEGOTIATE Responses, as described in [MS-SMB] section 2.2.4.5.2.1 or [MS-CIFS] section 2.2.4.52, are sent from the SMB File Service to the SMB File Client. This is described further in [MS-SMB] section 3.2.5.2. Once the negotiation has completed successfully, a connection exists between the SMB File Client and the SMB File Service.

3. On successful completion of the connection establishment, the SMB File Client attempts to establish a session between the SMB File Client and the SMB File Service by means of an iteration of one or more protocol messages.

4. One or more SMB_COM_SESSION_SETUP_ANDX Requests as described in [MS-SMB] section 2.2.4.6.1 or [MS-CIFS] section 2.2.4.53 are sent from the SMB File Client to the SMB File Service, resulting in SMB_COM_SESSION_SETUP_ANDX Responses, as described in [MS-SMB] section 2.2.4.6.2 or [MS-CIFS] section 2.2.4.53, until the identity of the SMB User has been authenticated and the establishment of the session authorized against the SMB File Service is completed. This is described further in [MS-SMB] section 3.2.4.2.4.

6.1.5.25.3 Final System State

- An authenticated and authorized session exists between the SMB File Client and the SMB File Service.
- There is an entry in the SMB File Server Connection List section 5.1.3.3.1 for the newly created connection.
- There is an entry in the SMB File Server Session List section 5.1.3.3.2 for the newly created session.

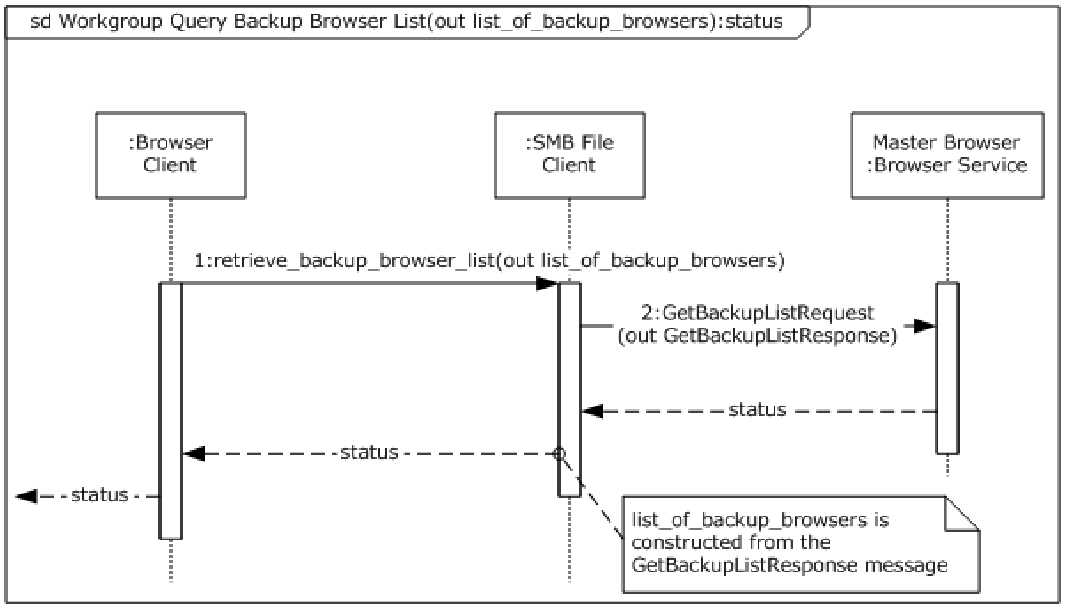
6.1.5.26 Workgroup Query Backup Browser List

The sequence described in this example details how the Browser Client retrieves a list of backup browsers from the Browser Service.

6.1.5.26.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.
- Master Browser is populated with a list of available backup browsers in the workgroup.

6.1.5.26.2 Sequence of Events



sd Workgroup Query Backup Browser List(out list_of_backup_browsers):status
:Browser Client
:SMB File Client
Master Browser :Browser Service
1:retrieve_backup_browser_list(out list_of_backup_browsers)
2:GetBackupListRequest (out GetBackupListResponse)
status
status
status
list_of_backup_browsers is constructed from the GetBackupListResponse message

Figure 42: Sequence diagram detail for Workgroup Query Backup Browser List

1. The Browser Client directs the SMB File Client to retrieve a list of backup browsers from the Master Browser. retrieve_backup_browser_list() is an example of an application specific message that the Browser Client uses to direct the SMB File Client to retrieve the list of backup browsers.

2. The SMB File Client uses GetBackupListRequest message as described in [MS-BRWS] section 2.2.4 to retrieve a list of backup browsers in the workgroup as returned in GetBackupListResponse (described in [MS-BRWS] section 2.2.5).

6.1.5.26.3 Final System State

The Browser Client has successfully retrieved a list of backup browsers in the workgroup.

6.1.5.27 Workgroup Query Computers List

The sequence described in this example details how the Browser Client retrieves a list of computers from the Browser Service.

6.1.5.27.1 Initial System State

- General requirements as set forth in section 4.2, System Assumptions and Preconditions.
- Requirements described in the "Initial System State" section of the sequence diagram referencing this sequence.
- Backup Browser is populated with a list of available computers in the workgroup.

6.1.5.27.2 Sequence of Events

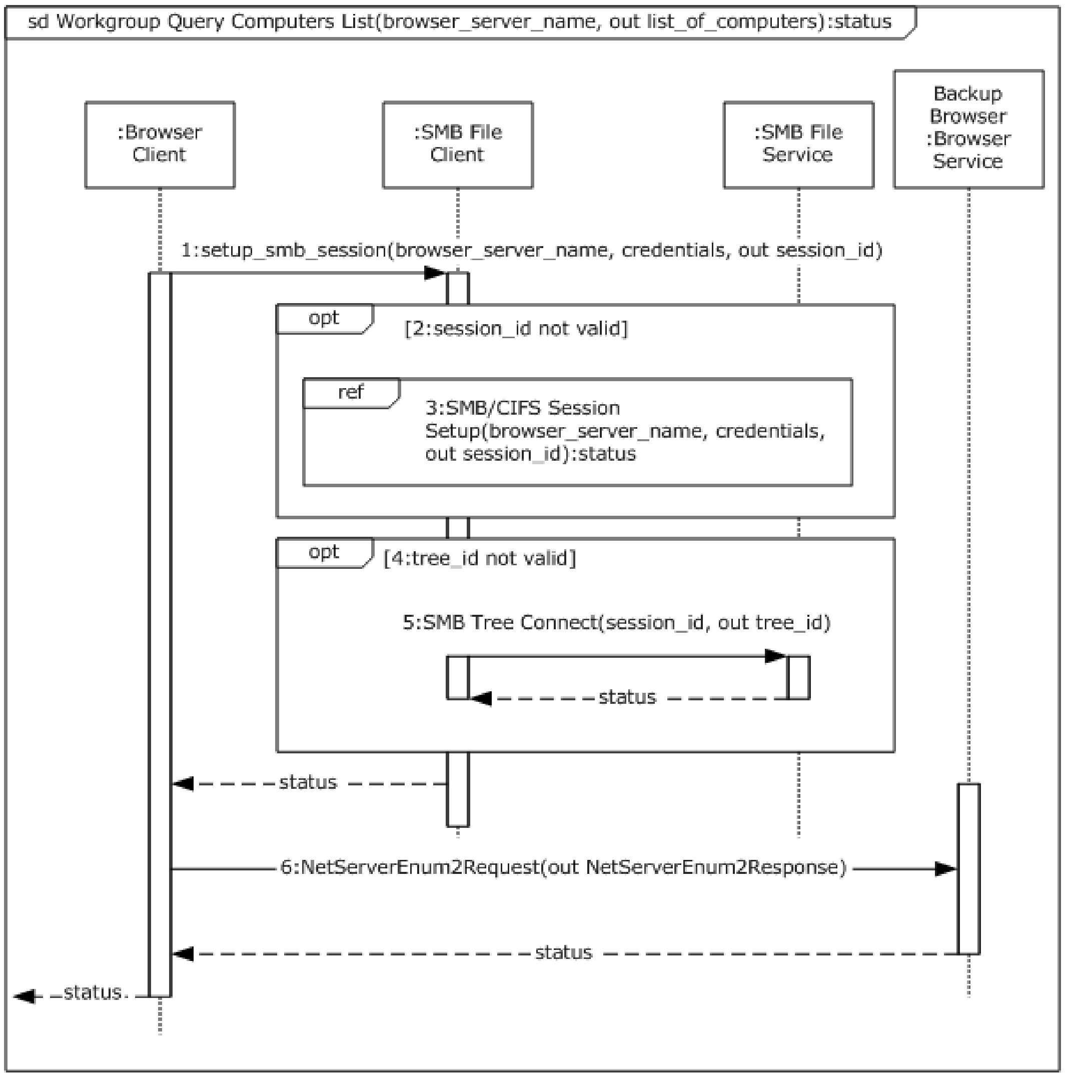

sd Workgroup Query Computers List(browser_server_name, out list_of_computers):status
:Browser Client
:SMB File Client
:SMB File Service
Backup Browser :Browser Service
1:setup_smb_session(browser_server_name, credentials, out session_id)
opt
[2:session_id not valid]
ref
3:SMB/CIFS Session Setup(browser_server_name, credentials, out session_id):status
opt
[4:tree_id not valid]
5:SMB Tree Connect(session_id, out tree_id)
status
status
6:NetServerEnum2Request(out NetServerEnum2Response)
status
status

Figure 43: Sequence diagram detail for Workgroup Query Computer List

1. The Browser Client directs the SMB File Client to establish a session to the SMB File Service on the computer represented by browser_server_name. setup_smb_session() is an example of an application specific message that the Browser Client uses to direct the SMB File Client to establish a session.

2. The SMB File Client examines some internal private state to determine whether or not there is an existing session between the SMB File Client and the SMB File Service that can be use to satisfy the request from the Browser Client. If a session is required, the SMB File Client proceeds to operation 3, otherwise it continues at operation 4.

3. The SMB File Client attempts to establish a session using the sequence SMB/CIFS Session Setup.

4. Once the SMB File Client has determined that a valid session has been established between the SMB File Client and the SMB File Service, it examines some internal private state to determine

whether or not there is a valid tree connection to SMB File Service. If there is no tree connection, SMB File Client proceeds to step 5, otherwise it continues to step 6. The process is further described in [MS-SMB2] section 3.2.4.2.4.

5. The SMB File Client attempts to establish a tree connection to the SMB File Service using an SMB_COM_TREE_CONNECT_ANDX Request (see [MS-SMB] section 2.2.4.7.1) using the previously obtained session identifier session_id. The SMB File Service sends an SMB_COM_TREE_CONNECT_ANDX Request Response (see [MS-SMB] section 2.2.4.7.2) with details of the tree connection, including an identifier tree_id that the SMB File Client can use to identify subsequent requests using the tree connection.

6. The Browser Client uses the NetServerEnum2Request message, as described in [MS-RAP] section 2.5.5.2.1, to retrieve a list of computers in the workgroup as returned in NetServerEnum2Response (as described in [MS-RAP] section 2.5.5.2.2).

6.1.5.27.3 Final System State

The Browser Client has successfully retrieved a list of computers in the workgroup.

6.2 Communication Details

Protocol layering was described in section 5.2. In general, each protocol may be transported by a number of lower-layer protocols, and the lower-layer protocol connection must be established before the given protocol connection can be established. Once a given protocol connection is established, any of that protocol's methods may be invoked. There are no method-dependent protocol transport restrictions. Consequently, the initiator of the transport connection determines what transport will be used.

The File Access Services System does not define any communication constraints or additional message types beyond those described in the specifications of the member protocols listed in section 2.2. Moreover, File Access Services Applications are free to use any of the remote methods supported by the various protocols. Consequently, it is not possible to meaningfully define a subset of remote protocol methods.

6.3 Transport Requirements

The File Access Services System does not define any transports or restrictions on transports beyond those described in the specifications of the member protocols listed in section 2.2 and summarized in this document in sections 5.1 and 6.1.5.25.

6.4 Timers

6.4.1 SMB Session Auto-Disconnect Timer

Purpose: Disconnect idle sessions either to the SMB File Service.

When Started: The timer is started as part of system initialization.

Reset or Restart Conditions: This timer is a repeating periodic timer and is restarted on timer expiration. The period is determined by Properties.AutoDisconnect as described in section 5.1.3.3.5.1.

Action on Timer Expiration: The idle session is disconnected. See [MS-SMB2] section 3.3.2.3.

6.4.2 SMB Oplock Break Response Wait Timer

Purpose: The period the SMB File Service uses to wait for the SMB File Client to respond to an Oplock Break Request.

When Started: This timer is started by the SMB File Service when an Oplock Break Request is sent to the SMB File Client previously granted the Oplock. The period is determined by Properties.OplockBreakResponseWait section 5.1.3.3.5.1.

Reset or Restart Conditions: The timer is a on a per-request timer and is not reset or restarted.

Action on Timer Expiration: See [MS-SMB2] section 3.3.2.1.

6.4.3 SMB Scavenger Thread Timer

Purpose: This timer is used to trigger one or more internal maintenance tasks in the SMB File Service such as monitor time-out requests, logging errors, updating server statistics, updating connection QoS parameters.

When Started: The timer is started as part of system initialization.

Reset or Restart Conditions: This timer is a repeating periodic timer and is re-started on timer expiration. The period is determined by Properties.ScavengerThreadTimeout section 5.1.3.3.5.1.

Action on Timer Expiration: When the timer expires, the SMB File Service flushes out any outstanding error log messages, updates the global server statistics, and queries the underlying transports for current link information, which is then used to update the SMB File Service QoS parameters.

6.5 Non-Timer Events

There are no non-timer events except for the ones described in the individual protocol Technical Documents.

6.6 Initialization and Reinitialization Procedures

There is no system initialization order requirement beyond what is described in the individual protocol Technical Documents.

6.7 Status and Error Returns

See the individual protocol Technical Documents.

7 Security

This section documents system-wide security issues that are not otherwise described in the Technical Documents (TDs) for the Member Protocols. It does not duplicate what is already in the Member Protocol TDs unless there is some unique aspect that applies to the system as a whole.

NFS and SMB file access protocols interoperate on a shared object store using security translation between the Windows security model as defined in [MS-AUTHSO] and AUTH_SYS security model as defined in [RFC5531]. Security on a file is stored in the object store and always adheres to the Windows security model. NFS file access protocols implement a translation from the native Windows security model to the AUTH_SYS RPC security model. This translation is achieved as described in [MSFT-UNIXPERM].

8 Appendix A: Product Behavior

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

- Windows 2000 operating system
- Windows 2000 Server operating system
- Windows 2000 Advanced Server 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 Server 2008 R2 operating system
- Windows 7 operating system

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

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

<1> Section 3.1.1: If the domain intends to support clients that only understand Windows NT 4.0-style domains, then the domain must have a unique NetBIOS name because Windows NT 4.0-style domains do not support DNS naming.

<2> Section 3.1.1: This style of domain is supported by all Windows operating systems for general purpose computers since Windows 2000.

<3> Section 3.3.2: In Windows, the Object Store is provided by a local file system, usually NTFS.

<4> Section 4.1: Also requires access to network services which support NetBIOS/NetBEUI if the system is to support older (pre-Windows 2000) SMB File Clients or File Services.

<5> Section 4.2: Windows Server 2003 R2 or later support S4U extensions.

<6> Section 5.2: The Remote Administration Protocol specified in [MS-RAP] has limited applicability to the File Access Services System and is only being used in support of clients of [MS-BRWS] discovering computers on the local network.

See [MS-RAP] section 1.6 and associated product behavior notes for details on the applicability of the Remote Administration Protocol and how it is used in the Windows operating systems within the context of the system described in this document.

<7> Section 5.2: The NFS and related protocols require TCP or UDP transport, but today they are used primarily with TCP as described in "Windows NFS Transport Selection". This note describes the Windows behavior for the NFS and related protocols for these areas:

- Mapping of NFS mode bits to Windows ACL
  - Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2 non-UUUA access:

    The NFS mode bits (or permissions mask) [RFC1094] section 2.3.5, and [RFC1813] section 2.5 are converted to a security descriptor as described in [MSFT-UNIXPERM].
  - Windows Server 2008 R2 Unmapped UNIX User Access (UUUA)

    If a Windows Server 2008 R2 NFS share is configured to enable Unmapped UNIX User Access and there is no existing mapping available to the NFS server (via either [MS-UNMP] or [RFC2307]) then the server will encode the owner, group, and mode of a file into a security descriptor directly using generated **SIDs**. The NFS server uses a specific sub-authority (SECURITY_NFS_ID_BASE_RID == 0x00000058) relative to the well known authority "NT Authority" (SECURITY_NT_AUTHORITY == {0,0,0,0,0,5}). The NFS server then uses further relative sub-authorities to build SIDs for different NfsTypes that represent the owner (0x00000001), the group (0x00000002), and the permissions mask (0x00000003) for the file. A further SID is also generated, which is used to store the other, or world access mask, within the security descriptor.

    Using the Windows string representation of a SID, this would follow this format:

    "<NTSecurityAuthority>-<SECURITY_NFS_ID_BASE_RID>-<NfsSidType>-<NfsSidValue>"

    Therefore, an example SIDs for an NFS user with a user identifier (user ID or UID) of 0x00000103 and a group identifier (group ID or GID) of 0x00000022 would be:

    - Owner SID "S-1-5-88-1-259"
    - Group SID "S-1-5-88-2-34"

    To construct a complete security descriptor, the NFS server generates a set of NFS-specific SIDs based on the UID, GID, and mode bits to be represented:

    - Owner SID based on the UID (for example, "S-1-5-88-1-<uid>")
    - Group SID based on the GID (for example, "S-1-5-88-2-<gid>")
    - Mode SID based on the UNIX mode bits (for example, "S-1-5-88-3-<mode>")
    - Other SID, a constant value (for example, "S-1-5-88-4")

    The NFS server then constructs a **discretionary access control list (DACL)** using a sequence of **Access Control Entries (ACEs)**:

    - A granted ACE is created to give the built-in SYSTEM account Full Control.
    - The owner, group, and other SIDs are then used to construct a set of grant and deny ACEs as described in [MSFT-UNIXPERM] to provide a Windows-style encoding of the UNIX mode bits.
    - A deny ACE is created using the generated SID for the permission mask, which directly encodes the UNIX mode for the file.

  - Additional ACEs may be inherited from the parent directory for the file being processed.

  Finally, the security descriptor is set as follows:

  - The Owner field is set to the generated owner SID.
  - The PrimaryGroup field is set to the generated group SID.
  - The DACL field is set to the generated DACL.
  - The **system access control list (SACL)** field may be set if inherited from the security descriptor for the parent directory.

- Read/execute equivalence between UNIX user access (UUUA)

  As NFS is stateless, there is no open operation where the NFS file client can indicate the intent behind the open. As a result, the NFS file service must perform access checks at the time of the READ request itself. NFS READ requests are unable to distinguish the purpose behind the request from the NFS file client; therefore the NFS file service cannot determine if the READ request is for a simple read of the file or whether the READ request is for the purposes of executing a program or script. The NFS file service will permit a READ request to complete if the user has either READ or EXECUTE access to the file.

  A consequence is that an SMB File Client file can set a security descriptor on a file, which allows FILE_EXECUTE but not FILE_READ_ACCESS access, which would allow an SMB User to execute a program file but not read that file.

- Delete access rights requirements when using UUUA with the NFS file service

  When the NFS file service is configured to use Unmapped UNIX User Access (UUUA), and the UUUA access rules are being used for a specific requesting user identity, strict UNIX access rights are applied so that any attempts to remove a name from the namespace using an NFS file client must have the ability to write to the directory containing that name. If an SMB File Client had used the SMB File Service to apply a security descriptor to a file that gave the NFS-specific SID for the UNIX user identity DELETE access to the file, that DELETE access would be ignored.

- Directory change notifications

  For file operations not involving namespace alteration or traversal, the default behavior of the NFS file service is to open files by their FileId (the file open request to the underlying object store uses the FILE_OPEN_BY_ID option). This bypasses the directory change mechanism used by the SMB File Service in order that changes to a file (for example, appending data to an existing file) might not trigger a change notification. The NFS behavior can be modified by means of the following registry key:

  HKLM\SYSTEM\CurrentControlSet\Services\NfsSvr\Parameters\FCCompat

  Setting the registry key to **DWORD** value 0x00000001 will force the NFS file service to always open a file by name. If the file has more than one name because of hard-links it is not predictable which name will be used.

- NFS and SMB symbolic link mechanisms

  The mechanisms used to represent symbolic links by the NFS file service and the SMB File Service are different.

A symbolic link set by an SMB File Client via the SMB File Service will be visible to an NFS file client via the NFS file service but will not be recognized as a symbolic link. The contents of the file may be accessible to the NFS file client via the NFS file service.

A symbolic link set by an NFS file client via the NFS file service will be visible to an SMB File Client via the SMB File Service but will not be recognized as a symbolic link. Attempts to read the file may succeed (depending upon access permissions) but the contents will not be that of the target of the symbolic link, but an encoding representing the target of the symbolic link. Any updates to the symbolic link by the SMB File Client via the SMB file server will invalidate the encoding and will invalidate the symbolic link to the NFS file client.

- NFS server automatic fallback to anonymous logon account or NULL_SID on AUTH_SYS identity mapping failure

  When responding to incoming requests, the NFS file service will attempt to map the AUTH_SYS style UID and GID to a recognized Windows account in order to obtain a SID and an access token for use when processing the incoming request. If the credentials cannot be converted to a recognized Windows User Account, the server will attempt to convert the credential to an anonymous account, or if that fails, the NFS file service will fall back to using the NULL SID based on the Windows security authority SECURITY_NULL_SID_AUTHORITY {0,0,0,0,0,0} with sub-authority SECURITY_NULL_RID 0x00000000.

  The anonymous account used by the NFS file service is "NT Authority\AnonymousLogon". This can be configured to an alternative account using the following registry key:

  HKLM\SYSTEM\CurrentControlSet\Services\NfsSvr\Parameters\UnmappedUnixUserUsername

  This key should be set to a REG_SZ string representation of a valid account name.

- Sub-auth package for server for NFS

  When the NFS file service is attempting to obtain an access token for a specific user account, the initial logon attempt is made using the authentication package named "MICROSOFT_AUTHENTICATION_PACKAGE_V1_0" with an NFS file service-specific Sub-Authentication Package with sub-authentication identity 0x000000AA.

- KeepInheritance

  By default, when creating a new file or directory the NFS file service does not propagate the Access Control List from the security descriptor of the parent directory to the security descriptor for the new file or directory. This behavior can be modified using the following registry key:

  HKLM\SYSTEM\CurrentControlSet\Services\NfsSvr\Parameters\KeepInheritance

  When this key is set to **DWORD** value 0x00000001 it will cause the NFS file service to propagate the access control entries from the discretionary access control list in the security descriptor for the parent directory to the appropriate location in the security descriptor for the new file. The propagation uses the normal Windows propagation rules described in [MS-DTYP] section 2.5.3.

- Windows NFS file client use of NLM protocol

  The NFS file client in the Windows implementation only uses NLMPROC4_SHARE/NLMPROC3_SHARE and NLMPROC4_UNSHARE/NLMPROC3_UNSHARE as defined in [RFC1813] section 6.2 and [C702].

- Windows NFS file client use of NSM protocol

The NFS file client in the Windows implementation does not use NSM protocol at all.

- Windows NFS file service use of NSM protocol

  The NFS file service in the Windows implementation always returns a SUNRPC success code but takes no other action for the SM_SIMU_CRASH procedure defined in [C702].

- Windows Portmap service

  The Windows implementation of the portmapper protocol [RFC1057] limits the behavior of the portmap service.

  1. Windows implementation of the RPCBIND protocol was introduced in Windows Server 2008 and is not present in Windows Server 2003 R2.

  2. Windows implementation of RPCBIND protocol version 3 and version 4 only implements the following set of procedures: RPCBPROC3_NULL/RPCBPROC4_NULL, RPCBPROC3_SET/RPCBPROC4_SET, RPCBPROC3_UNSET/RPCBPROC4_UNSET, RPCBPROC3_GETADDR/RPCBPROC4_GETADDR, RPCBPROC3_DUMP/RPCBPROC4_DUMP, and RPCBPROC4_GETADDRLIST as defined in [RFC1833] section 2.1.

  3. By default, the portmap service will only process PMAP_CALLIT requests made using UDP (IPv4 or IPv6), to either the NULL procedure call for any registered program version, or a NIS service registered with an IPv4, UDP port for program version 2. This behavior can be modified by setting the following REG_SZ registry key:

     HKLM\SYSTEM\CurrentControlSet\Services\Portmap\Parameters\IndirectCallStyle

     as follows:

     - The string "NULL" for the default behavior

     - The string "NONE" to stop all responses except to the NIS program version 2 with an IPv4 UDP port registration

     - The string "ANY" to forward the request to any program with a registration for an UDP IPv4 port

  4. PMAP_SET, PMAP_UNSET, RPCB_SET, and RPCB_UNSET requests will be rejected unless they are initiated from an address recognized as local to the computer. Rejected requests are silently dropped.

  5. PMAP_UNSET and RPCB_UNSET requests using AUTH_SYS credentials will succeed only if the UID or GID supplied matches those used for the registration of that program. If either the UID or GID does not match, then the portmap service will return an RPC success message with a PMAP_UNSET result of "FALSE".

  6. PMAP_UNSET and RPCB_UNSET requests using AUTH_NULL credentials are not affected by any UID and GID used in the program registration.

- Windows NFS transport selection

  - The NFS file client and NFS file service in the Windows implementation use the TCP protocol by default. When TCP is unavailable, the connection attempt will be made using the UDP protocol.

  - The NFS file client uses NFS version 3 by default. When NFS version 3 is unavailable at the NFS file service, the NFS file client attempts to use NFS version 2.

<8> Section 5.3.1: Beginning with Windows 7, Remote Administration Protocol is no longer enabled for these methods.

<9> Section 5.3.1: Windows Server 2008 and Windows Vista provided a completely new version of SMB, referred to as SMB2. It is the default file server protocol used to communicate between the current Windows versions, including Windows 7. It contains several new features, such as advanced pipelining, symbolic links, a new form of oplocks called leasing, support for hosted caching, durable and resilient handles, and improved scalability of basic structures like numbers of shares, users, and open files.

<10> Section 5.4: This applies to Windows 2000 client and server.

<11> Section 5.4: The following also apply to DFS Service (on domain controller (DC)): Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

9 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

10 Index