

# Open Specifications

Article • 08/01/2023

Through the Open Specifications program, Microsoft is helping developers open new opportunities by making technical documents related to interoperability for certain popular Microsoft products available to view and download at no charge.

Although the Open Specifications technical documents are freely available, many of them include patented inventions. Some of these patents are available at no charge under the [Open Specifications Promise](#) or the [Microsoft Community Promise](#). The remaining patents are available through various licensing programs. For more information, please visit the [Microsoft Open Specifications Dev Center](#) ↗ website or send an email message to the [IP Licensing Team](#).

 <a href="#">Protocols</a>	Microsoft publishes technical documents for protocols that are implemented in Windows client (including .NET Framework) and Windows Server (collectively Windows), Office, SharePoint Products and Technologies, Exchange Server, and Microsoft SQL Server and are used to communicate with other Microsoft software products.
 <a href="#">Computer Languages</a>	Microsoft publishes technical documents for the VBA programming language and Extensible Application Markup Language (XAML).
 <a href="#">Standards Support</a>	Microsoft publishes technical documents that describe support for specific standards implemented in Exchange Server and Outlook; Internet Explorer; OData; Microsoft SQL Server; Windows WordPad; and Word, Excel, and PowerPoint.
 <a href="#">Data Portability</a>	Microsoft publishes technical documents for the file formats created by Word, Excel, PowerPoint, and Outlook and by SQL Server. Additionally, it publishes technical documents that describe how user-created data in SQL Server can be extracted for use in other software products.

Microsoft revises the technical documents regularly and, particularly, in connection with the release of significant product updates and new versions.

## Q&A and Blogs

 ↗	<a href="#">Exchange Server Open Specifications and Standards Support - Microsoft Q&amp;A ↗</a> Home to technical questions and answers about Exchange Server Open Specifications and Standards Support documents.
 ↗	<a href="#">Office Open Specifications, Standards Support, and File Formats - Microsoft Q&amp;A ↗</a> Home to technical questions and answers about Office Open Specifications, Standards Support, and File Format documents.
 ↗	<a href="#">SharePoint Server Open Specifications - Microsoft Q&amp;A ↗</a> Home to technical questions and answers about SharePoint Server Open Specifications documents.
 ↗	<a href="#">SQL Server Open Specifications, Standards Support and Data Portability - Microsoft Q&amp;A ↗</a> Home to technical questions and answers about SQL Server Open Specifications, Standards Support and Data Portability documents.
 ↗	<a href="#">Windows Open Specifications and Standards Support - Microsoft Q&amp;A ↗</a> Home to technical questions and answers about Windows Open Specifications and Standards Support documents.
 ↗	<a href="#">Open Specifications Blogs ↗</a> These blogs, authored by the engineers who support the Open Specifications documents, provide a different venue for further discussion of those documents.

## Also in this section

- [Programs ↗](#)

# Protocols

Article • 08/01/2023

The Protocols section provides detailed Open Specifications technical documents for certain protocols that are implemented in Exchange Server, Office, SharePoint Products and Technologies, Microsoft SQL Server, Windows client (including .NET Framework) and Windows Server collectively published under Windows Protocols and are used to communicate with other Microsoft software products.

Although the Open Specifications technical documents are freely available, many of them include patented inventions. Some of these patents are available at no charge under the [Open Specifications Promise](#) or the [Microsoft Community Promise](#). The remaining patents are available through various licensing programs. For more information, please visit the Microsoft [Open Specifications Dev Center](#) website or send an email message to the [IP Licensing Team](#).

## Q&A and Blogs

 <a href="#">Exchange Server Open Specifications and Standards Support - Microsoft Q&amp;A ↗</a>
Home to technical questions and answers about Exchange Server Open Specifications and Standards Support documents.
 <a href="#">Office Open Specifications, Standards Support, and File Formats - Microsoft Q&amp;A ↗</a>
Home to technical questions and answers about Office Open Specifications, Standards Support, and File Format documents.
 <a href="#">SharePoint Server Open Specifications - Microsoft Q&amp;A ↗</a>
Home to technical questions and answers about SharePoint Server Open Specifications documents.
 <a href="#">SQL Server Open Specifications, Standards Support and Data Portability - Microsoft Q&amp;A ↗</a>
Home to technical questions and answers about SQL Server Open Specifications, Standards Support and Data Portability documents.
 <a href="#">Windows Open Specifications and Standards Support - Microsoft Q&amp;A ↗</a>
Home to technical questions and answers about Windows Open Specifications and Standards Support documents.
 <a href="#">Open Specifications Blogs ↗</a>

These blogs, authored by the engineers who support the Open Specifications documents, provide a different venue for further discussion of those documents.

# Explore the Protocols Documentation

- [Exchange Server Protocols](#)
- [Office Protocols](#)
- [SharePoint Products and Technologies Protocols](#)
- [Microsoft SQL Server Protocols](#)
- [Windows Protocols](#)

Downloads	Related Sections	Sites
<a href="#">Exchange Server Protocols .zip file ↗ (120+ MB)</a>	<a href="#">Document Programs ↗</a>	<a href="#">Open Specifications Dev Center ↗</a>
<a href="#">Microsoft Office Protocols .zip file ↗ (80+ MB)</a>	<a href="#">Computer Languages</a>	
<a href="#">SharePoint Products and Technologies Protocols .zip file ↗ (200+ MB)</a>	<a href="#">Standards Support</a>	
<a href="#">Microsoft SQL Server Protocols .zip file ↗ (50+ MB)</a>	<a href="#">Data Portability</a>	
<a href="#">Windows Protocols .zip file ↗ (500+ MB)</a>		

# Windows Protocols

Article09/12/2022

This page and associated content may be updated frequently. We recommend you subscribe to the [RSS feed](#) to receive update notifications.



This documentation contains detailed technical specifications for Microsoft protocols that are implemented and used by Windows to interoperate or communicate with other Microsoft products. It also contains technical specifications for extensions to industry-standard and other published protocols that are used by Windows. In addition, the documentation includes a set of companion technology overview and reference documents that supplement the technical specifications with conceptual background, overviews of inter-protocol relationships and interactions, and technical reference information.

Although the Open Specifications technical documents are freely available, many of them include patented inventions. Some of these patents are available at no charge under the [Open Specifications Promise](#) or the [Microsoft Community Promise](#). The remaining patents are available through various licensing programs. For more information, please visit the [Microsoft Open Specifications Dev Center](#) or send an email message to the [IP Licensing Team](#).

## Explore Windows Protocols Documentation

<a href="#">What's New and Changed in Windows Documents</a>
Provides information about and links to new and updated protocol documents that contain details about the most recently released versions of Windows Client and Windows Server operating systems.
<a href="#">Windows Protocols Preview Documents</a>
Provides preliminary versions of new or updated Open Specifications technical documents for community review and feedback.
<a href="#">Windows Protocols Errata</a>
Provides clarifications and adjustments of content issues in published versions of protocol documents that could impact an implementation.

▪ <a href="#">Windows Protocols Overview Documents</a>
Provides information about the protocols and other technologies that are included in the Windows Protocols documentation set and the relationships among those technologies.
▪ <a href="#">Windows Protocols Technical Documents</a>

## Forums and Blogs

 <a href="#">Open Specifications Forums</a> ↗
These user forums are available to answer technical questions about the Open Specifications documents.
 <a href="#">Open Specifications Blogs</a> ↗

These blogs, authored by the engineers who support the Open Specifications documents, provide a different venue for further discussion of those documents.

### [Windows Interoperability Developer Blog](#)

This blog is updated with the latest information on releases of the Open Specifications documents.

Downloads	Sites
<a href="#">Windows Protocols PDF .zip file</a> ↗ (600+ MB)	<a href="#">Open Specifications Developer Center</a> ↗
<a href="#">Standards Support Downloads</a>	<a href="#">FAQ: Network Captures for Technology Overview Documents</a> ↗  <a href="#">Windows Protocols Archived Forums</a> ↗

# Technical Documents

Article03/13/2023

This section provides information about the technical specifications that are contained in the Windows Protocols documentation set.

For preview or pre-release versions of the technical specifications, see [Preview Documents](#).

**Note** The inter-document links in a PDF version of a technical specification document are functional only if all the cross-referenced documents are saved to the same local directory folder. An error message appears if you click a link that references a PDF document that is not located in the same folder (when viewing via your local hard drive) or is part of a different download (when viewing online). To save a complete set of PDF files to the same folder, download the [Windows Protocols .zip file](#). This is a large file and can take a few minutes to download.

Specification	Description
[MC-BUP]: Background Intelligent Transfer Service (BITS) Upload Protocol	Specifies the Background Intelligent Transfer Service (BITS) Upload Protocol, which is used to upload large entities from a client to a server over networks with frequent disconnections, and to send notifications from the server to a server application about the availability of the uploaded entities.  <a href="#">Click here to view this version of the [MC-BUP] PDF.</a>
[MC-CCFG]: Server Cluster: Configuration (ClusCfg) Protocol	Specifies the Server Cluster: Configuration (ClusCfg) Protocol, which enables users to restore a node that is no longer a configured member of a failover cluster back to its pre-cluster installation state.  <a href="#">Click here to view this version of the [MC-CCFG] PDF.</a>
[MC-COMQC]: Component Object Model Plus (COM+) Queued Components Protocol	Specifies the Component Object Model Plus (COM+) Queued Components Protocol, which is used for persisting method calls made on COM+ objects in such a way that they can later be played back and executed.  <a href="#">Click here to view this version of the [MC-COMQC] PDF.</a>
[MC-CSIDL]: Conceptual Schema Definition File Format	Specifies the Conceptual Schema Definition File Format, which defines some well-known primitive types, such as Edm.String, that are used as the building blocks for structural types like Entity Types and Complex Types.  <a href="#">Click here to view this version of the [MC-CSIDL] PDF.</a>

Specification	Description
<a href="#">[MC-DPL4CS]: DirectPlay 4 Protocol: Core and Service Providers</a>	<p>Specifies DirectPlay 4 Protocol: Core and Service Providers. This protocol enables the implementation of functions to enumerate hosted game sessions and players, to add and remove game players, and to interchange data between game instances.</p> <p><a href="#">Click here to view this version of the [MC-DPL4CS] PDF.</a> ↗</p>
<a href="#">[MC-DPL4R]: DirectPlay 4 Protocol: Reliable</a>	<p>Specifies the DirectPlay 4 Protocol: Reliable, which describes functionality related to the reliable delivery of DirectPlay 4 messages and provides throttling for applications that use DirectPlay 4.</p> <p><a href="#">Click here to view this version of the [MC-DPL4R] PDF.</a> ↗</p>
<a href="#">[MC-DPL8CS]: DirectPlay 8 Protocol: Core and Service Providers</a>	<p>Specifies the DirectPlay 8 Protocol: Core and Service Providers, which creates and manages game sessions over existing datagram protocols such as UDP.</p> <p><a href="#">Click here to view this version of the [MC-DPL8CS] PDF.</a> ↗</p>
<a href="#">[MC-DPL8R]: DirectPlay 8 Protocol: Reliable</a>	<p>Specifies the DirectPlay 8 Protocol: Reliable, which provides mixed, not reliable, and reliable messages over existing datagram protocols such as the User Datagram Protocol (UDP).</p> <p><a href="#">Click here to view this version of the [MC-DPL8R] PDF.</a> ↗</p>
<a href="#">[MC-DPLHP]: DirectPlay 8 Protocol: Host and Port Enumeration</a>	<p>Specifies the DirectPlay 8 Protocol: Host and Port Enumeration, which enables a DirectPlay 8 client application to discover one or more DirectPlay 8 server applications.</p> <p><a href="#">Click here to view this version of the [MC-DPLHP] PDF.</a> ↗</p>
<a href="#">[MC-DPLNAT]: DirectPlay 8 Protocol: NAT Locator</a>	<p>Specifies the DirectPlay 8 Protocol: NAT Locator, which provides extensions to the DirectPlay 8 Core and Service Providers Protocol (as specified in [MC-DPL8CS]) to improve Network Address Translation (NAT) support.</p> <p><a href="#">Click here to view this version of the [MC-DPLNAT] PDF.</a> ↗</p>
<a href="#">[MC-DPLVP]: DirectPlay Voice Protocol</a>	<p>Specifies the DirectPlay Voice Protocol, which is used to provide voice communications for applications that use the DirectPlay protocol to communicate.</p> <p><a href="#">Click here to view this version of the [MC-DPLVP] PDF.</a> ↗</p>
<a href="#">[MC-DRT]: Distributed Routing Table (DRT) Version 1.0</a>	<p>Specifies the Distributed Routing Table (DRT) Version 1.0 protocol, which is used to maintain a network of nodes (referred to as a cloud) and to resolve keys to their endpoint information when requested by a node within the cloud.</p> <p><a href="#">Click here to view this version of the [MC-DRT] PDF.</a> ↗</p>

Specification	Description
[MC-DTCXA]: MSDTC Connection Manager: OleTx XA Protocol	<p>Specifies the MSDTC Connection Manager: OleTx Transaction Protocol, which describes the extensions that support XA [XOPEN-DTP]-compliant software components in an OleTx distributed transaction processing environment.</p> <p><a href="#">Click here to view this version of the [MC-DTCXA] PDF.</a> ↗</p>
[MC-EDMX]: Entity Data Model for Data Services Packaging Format	<p>Specifies the Entity Data Model for Data Services Packaging Format (EDMX), which is an XML-based file format that serves as the packaging format for the service metadata of a data service.</p> <p><a href="#">Click here to view this version of the [MC-EDMX] PDF.</a> ↗</p>
[MC-IISA]: Internet Information Services (IIS) Application Host COM Protocol	<p>Specifies the Internet Information Services (IIS) Application Host COM Protocol, which provides read/write access to administrative configuration data that is located on a remote server.</p> <p><a href="#">Click here to view this version of the [MC-IISA] PDF.</a> ↗</p>
[MC-MQAC]: Message Queuing (MSMQ): ActiveX Client Protocol	<p>Specifies the Message Queuing (MSMQ): ActiveX Client Protocol, which is a collection of Distributed Component Object Model (DCOM) [MS-DCOM] interfaces that expose message queuing functionality for use by client applications.</p> <p><a href="#">Click here to view this version of the [MC-MQAC] PDF.</a> ↗</p>
[MC-MQSRM]: Message Queuing (MSMQ): SOAP Reliable Messaging Protocol (SRMP)	<p>Specifies the Message Queuing (MSMQ): SOAP Reliable Messaging Protocol (SRMP), which defines a mechanism for reliably transferring messages between two message queues that are located on two different hosts.</p> <p><a href="#">Click here to view this version of the [MC-MQSRM] PDF.</a> ↗</p>
[MC-NBFS]: .NET Binary Format: SOAP Data Structure	<p>Specifies the SOAP data structure for the .NET Binary Format for XML. This structure uses the XML data structure format [MC-NBFX], but specifies the set of strings to which a producer and consumer can refer.</p> <p><a href="#">Click here to view this version of the [MC-NBFS] PDF.</a> ↗</p>
[MC-NBFSE]: .NET Binary Format: SOAP Extension	<p>Specifies the SOAP extension for the .NET Binary Format for XML. This SOAP extension is a new format built by extending the format specified in [MC-NBFS]; it provides a context under which strings may be transmitted once and referred to by subsequent documents in order to reduce the size of the documents.</p> <p><a href="#">Click here to view this version of the [MC-NBFSE] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MC-NBFX]: .NET Binary Format: XML Data Structure</a>	<p>Specifies the XML data structure for the .NET Binary Format for XML. This format can represent many XML documents, as specified in [XML1.0]. The purpose of the format is to reduce the processing costs associated with XML documents by encoding an XML document in fewer bytes than the same document encoded in UTF-8, as specified in [RFC2279].</p> <p><a href="#">Click here to view this version of the [MC-NBFX] PDF.</a></p>
<a href="#">[MC-NETCEX]: .NET Context Exchange Protocol</a>	<p>Specifies the .NET Context Exchange Protocol, which defines a message syntax for identifying context that is shared between a client and a server, and a protocol for establishing that context.</p> <p><a href="#">Click here to view this version of the [MC-NETCEX] PDF.</a></p>
<a href="#">[MC-NMF]: .NET Message Framing Protocol</a>	<p>Specifies the .NET Message Framing Protocol, which defines a mechanism for framing messages. While this is primarily aimed at framing SOAP messages, the protocol can be used to frame other message types as well.</p> <p><a href="#">Click here to view this version of the [MC-NMF] PDF.</a></p>
<a href="#">[MC-NPR]: .NET Packet Routing Protocol</a>	<p>Specifies the .NET Packet Routing Protocol, which defines a SOAP message header to indicate that a message can safely be treated as a packet or datagram.</p> <p><a href="#">Click here to view this version of the [MC-NPR] PDF.</a></p>
<a href="#">[MC-PRCH]: Peer Channel Protocol</a>	<p>Specifies the Peer Channel Protocol, which is used for broadcasting messages over a virtual network of cooperating nodes.</p> <p><a href="#">Click here to view this version of the [MC-PRCH] PDF.</a></p>
<a href="#">[MC-PRCR]: Peer Channel Custom Resolver Protocol</a>	<p>Specifies the Peer Channel Custom Resolver Protocol, which is used for storage and retrieval of endpoint information of clients with access to a known service.</p> <p><a href="#">Click here to view this version of the [MC-PRCR] PDF.</a></p>
<a href="#">[MC-SMP]: Session Multiplex Protocol</a>	<p>Specifies the Session Multiplex Protocol, which provides session management capabilities between a database client and a database server. This protocol enables multiple logical client connections to connect to a single server over a single physical connection.</p> <p><a href="#">Click here to view this version of the [MC-SMP] PDF.</a></p>
<a href="#">[MC-SQLR]: SQL Server Resolution Protocol</a>	<p>Specifies the SQL Server Resolution Protocol, which facilitates connectivity to a database server.</p> <p><a href="#">Click here to view this version of the [MC-SQLR] PDF.</a></p>

Specification	Description
<a href="#">[MS-ABTP]: Automatic Bluetooth Pairing Protocol</a>	<p>Specifies the Automatic Bluetooth Pairing Protocol, which facilitates the establishment of a secure, trusted Bluetooth pairing relationship between two devices without requiring any user interaction at the time of pairing.</p> <p><a href="#">Click here to view this version of the [MS-ABTP] PDF.</a></p>
<a href="#">[MS-ADA1]: Active Directory Schema Attributes A-L</a>	<p>Specifies the Active Directory Schema Attributes A-L, which contains a partial list of the objects that exist in the Active Directory schema (attributes beginning with A - L).</p> <p><a href="#">Click here to view this version of the [MS-ADA1] PDF.</a></p>
<a href="#">[MS-ADA2]: Active Directory Schema Attributes M</a>	<p>Specifies the Active Directory Schema Attributes M, which contains a partial list of the objects that exist in the Active Directory schema (attributes beginning with M).</p> <p><a href="#">Click here to view this version of the [MS-ADA2] PDF.</a></p>
<a href="#">[MS-ADA3]: Active Directory Schema Attributes N-Z</a>	<p>Specifies the Active Directory Schema Attributes N-Z, which contains a partial list of the objects that exist in the Active Directory schema (attributes beginning with N through Z).</p> <p><a href="#">Click here to view this version of the [MS-ADA3] PDF.</a></p>
<a href="#">[MS-ADCAP]: Active Directory Web Services: Custom Action Protocol</a>	<p>Specifies the Active Directory Web Services: Custom Action Protocol, used for directory access in identity management and topology management. This protocol enables the transition of client applications that are currently using non-web services protocols for managing information held in directory services to instead use Web services protocols.</p> <p><a href="#">Click here to view this version of the [MS-ADCAP] PDF.</a></p>
<a href="#">[MS-ADDM]: Active Directory Web Services: Data Model and Common Elements</a>	<p>Specifies the Active Directory Web Services: Data Model and Common Elements. This protocol contains an XML data model and other protocol components (such as the definition of an XPath 1.0-derived selection language) that are used in various protocols that belong to the set of Active Directory Web Services protocols.</p> <p><a href="#">Click here to view this version of the [MS-ADDM] PDF.</a></p>
<a href="#">[MS-ADFSOAL]: Active Directory Federation Services OAuth Authorization Code Lookup Protocol</a>	<p>Specifies the Active Directory Federation Services OAuth Authorization Code Lookup Protocol, which is used to find the issuing server of an access token for an OAuth authorization code.</p> <p><a href="#">Click here to view this version of the [MS-ADFSOAL] PDF.</a></p>

Specification	Description
<a href="#">[MS-ADFSPIP]: Active Directory Federation Services and Proxy Integration Protocol</a>	<p>Specifies the Active Directory Federation Services Proxy and Web Application Proxy Integration Protocol. This protocol integrates Active Directory Federation Services with an authentication and application proxy to enable access to services located inside the boundaries of the corporate network for clients that are located outside of that boundary.</p> <p><a href="#">Click here to view this version of the [MS-ADFSPIP] PDF.</a></p>
<a href="#">[MS-ADFSPP]: Active Directory Federation Service (AD FS) Proxy Protocol</a>	<p>Specifies the Federation Service Proxy Protocol, which is used by a security token service (STS) proxy to obtain configuration data about an STS in order to assist users in selecting an acceptable security realm from which to obtain a security token.</p> <p><a href="#">Click here to view this version of the [MS-ADFSPP] PDF.</a></p>
<a href="#">[MS-ADFSWAP]: Active Directory Federation Service (AD FS) Web Agent Protocol</a>	<p>Specifies the Federation Service Web Agent Protocol, which is used by a Web service (WS) resource to obtain configuration data about a security token service (STS) in order to validate tokens from that STS using the protocol defined in [MS-MWBF].</p> <p><a href="#">Click here to view this version of the [MS-ADFSWAP] PDF.</a></p>
<a href="#">[MS-ADLS]: Active Directory Lightweight Directory Services Schema</a>	<p>Specifies the Active Directory Lightweight Directory Services Schema, which contains a list of the objects that exist in the Active Directory Lightweight Directory Services schema.</p> <p><a href="#">Click here to view this version of the [MS-ADLS] PDF.</a></p>
<a href="#">[MS-ADSC]: Active Directory Schema Classes</a>	<p>Specifies the Active Directory Schema Classes, which contains a partial list of objects that exist in the Active Directory schema.</p> <p><a href="#">Click here to view this version of the [MS-ADSC] PDF.</a></p>
<a href="#">[MS-ADTG]: Remote Data Services (RDS) Transport Protocol</a>	<p>Specifies the Remote Data Services (RDS) Transport Protocol, an HTTP request/response protocol that facilitates remote method definition and invocation, method definitions for executing database commands and for synchronizing database results, and definition of a record format for encoding of database results.</p> <p><a href="#">Click here to view this version of the [MS-ADTG] PDF.</a></p>
<a href="#">[MS-ADTS]: Active Directory Technical Specification</a>	<p>Specifies the core functionality of Active Directory. Active Directory extends and provides variations of the Lightweight Directory Access Protocol (LDAP).</p> <p><a href="#">Click here to view this version of the [MS-ADTS] PDF.</a></p>

Specification	Description
<a href="#">[MS-AIPS]: Authenticated Internet Protocol</a>	<p>Specifies the Authenticated Internet Protocol. This protocol supports a more generalized authentication exchange than the Internet Key Exchange Protocol and provides the optimizations in key exchange and policy discoverability.</p> <p><a href="#">Click here to view this version of the [MS-AIPS] PDF.</a> ↗</p>
<a href="#">[MS-APDS]: Authentication Protocol Domain Support</a>	<p>Specifies Authentication Protocol Domain Support, which is the communication process between a server and a domain controller that uses Netlogon interfaces to complete an authentication sequence.</p> <p><a href="#">Click here to view this version of the [MS-APDS] PDF.</a> ↗</p>
<a href="#">[MS-ASP]: ASP.NET State Server Protocol</a>	<p>Specifies the ASP.NET State Server Protocol, which is a contract for transmitting session state data between a client and a state server.</p> <p><a href="#">Click here to view this version of the [MS-ASP] PDF.</a> ↗</p>
<a href="#">[MS-AZMP]: Authorization Manager (AzMan) Policy File Format</a>	<p>Specifies the Authorization Manager (AzMan) Policy File Format, which defines the XML structure of AzMan policy files. These files are used by the Microsoft Management Console (MMC) AzMan snap-in and the authorization manager runtime.</p> <p><a href="#">Click here to view this version of the [MS-AZMP] PDF.</a> ↗</p>
<a href="#">[MS-BDSRR]: Business Document Scanning: Scan Repository Capabilities and Status Retrieval Protocol</a>	<p>Specifies the Business Document Scanning: Scan Repository Capabilities and Status Retrieval Protocol, which is used to query a server for the capabilities and status of the scan repository.</p> <p><a href="#">Click here to view this version of the [MS-BDSRR] PDF.</a> ↗</p>
<a href="#">[MS-BGPP]: Border Gateway Protocol (BGP) Profile</a>	<p>Specifies Border Gateway Protocol (BGP) Profile a dynamic routing protocol that automatically learns routes between sites that are connected using site-to-site VPN connections and clarifies what portions of [RFC1997] and [RFC4271] are not supported.</p> <p><a href="#">Click here to view this version of the [MS-BGPP] PDF.</a> ↗</p>
<a href="#">[MS-BKRP]: BackupKey Remote Protocol</a>	<p>Specifies the BackupKey Remote Protocol. This protocol encrypts secret values (such as cryptographic keys) so they can be backed up to storage that is not specially protected, and enables decryption of such values if recovery is necessary.</p> <p><a href="#">Click here to view this version of the [MS-BKRP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-BKUP]: Microsoft NT Backup File Structure</a>	<p>Specifies the Microsoft NT Backup File Structure protocol, which describes the network format of the Windows NT backup file format and its constituent structures that may be used in other protocols.</p> <p><a href="#">Click here to view this version of the [MS-BKUP] PDF.</a> ↗</p>
<a href="#">[MS-BPAU]: Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Authentication Protocol</a>	<p>Specifies the Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Authentication Protocol. This protocol provides authentication for computers in an Active Directory domain in support of the BITS Peer-Caching Content Retrieval Protocol ([MS-BPCR]).</p> <p><a href="#">Click here to view this version of the [MS-BPAU] PDF.</a> ↗</p>
<a href="#">[MS-BPCR]: Background Intelligent Transfer Service (BITS) Peer-Caching: Content Retrieval Protocol</a>	<p>Specifies the Background Intelligent Transfer Service (BITS) Peer-Caching: Content Retrieval Protocol, which is one of the family of protocols that implements a distributed URL cache known as ""BITS peer-caching"". Other protocols in the family are used to discover potential peers and to authenticate them.</p> <p><a href="#">Click here to view this version of the [MS-BPCR] PDF.</a> ↗</p>
<a href="#">[MS-BPDP]: Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Discovery Protocol</a>	<p>Specifies the Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Discovery Protocol, which is used to locate hosts in a domain that supports the URL-caching protocol implemented by BITS.</p> <p><a href="#">Click here to view this version of the [MS-BPDP] PDF.</a> ↗</p>
<a href="#">[MS-BRWS]: Common Internet File System (CIFS) Browser Protocol</a>	<p>Specifies the Common Internet File System (CIFS) Browser Protocol, which updates all backup browser servers with the contents of the response to a NetServerEnum2 request and shares the processing load of enumerating the services available in the network across different servers.</p> <p><a href="#">Click here to view this version of the [MS-BRWS] PDF.</a> ↗</p>
<a href="#">[MS-BRWSA]: Common Internet File System (CIFS) Browser Auxiliary Protocol</a>	<p>Specifies the Common Internet File System (CIFS) Browser Auxiliary Protocol, which is used by the master browser server to query configuration information for the domains from the domain master browser server.</p> <p><a href="#">Click here to view this version of the [MS-BRWSA] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-CAPR]: Central Access Policy Identifier (ID) Retrieval Protocol</a>	<p>Specifies the Central Access Policy ID Retrieval Protocol, which allows administrative applications to retrieve the set of central access policies deployed on remote computers.</p> <p><a href="#">Click here to view this version of the [MS-CAPR] PDF.</a></p>
<a href="#">[MS-CBCP]: Callback Control Protocol</a>	<p>Specifies the Callback Control Protocol, which provides a standard method for transporting multi-protocol datagrams over point-to-point links.</p> <p><a href="#">Click here to view this version of the [MS-CBCP] PDF.</a></p>
<a href="#">[MS-CDP]: Connected Devices Platform Protocol Version 3</a>	<p>Specifies the Connected Devices Platform Protocol Version 3. This protocol provides a discovery system to authenticate and verify users and devices, as well as providing a message exchange between devices. It provides a transport-agnostic means of building connections among all of a user's devices, whether available through the cloud or through direct physical presence.</p> <p><a href="#">Click here to view this version of the [MS-CDP] PDF.</a></p>
<a href="#">[MS-CER]: Corporate Error Reporting Version 1.0 Protocol</a>	<p>Specifies the Corporate Error Reporting Version 1.0 Protocol, which enables an organization to copy error reports from a set of client machines to a CER file share on a specified Server Message Block (SMB) Protocol file server with additional configuration options.</p> <p><a href="#">Click here to view this version of the [MS-CER] PDF.</a></p>
<a href="#">[MS-CER2]: Corporate Error Reporting V.2 Protocol</a>	<p>Specifies the Corporate Error Reporting V.2 Protocol, which enables enterprise computing sites to manage all error reporting information within the organization.</p> <p><a href="#">Click here to view this version of the [MS-CER2] PDF.</a></p>
<a href="#">[MS-CFB]: Compound File Binary File Format</a>	<p>Specifies the Compound File Binary File Format, a general-purpose file format that provides a file-system-like structure within a file for the storage of arbitrary, application-specific streams of data.</p> <p><a href="#">Click here to view this version of the [MS-CFB] PDF.</a></p>
<a href="#">[MS-CHAP]: Extensible Authentication Protocol Method for Microsoft Challenge Handshake Authentication Protocol (CHAP)</a>	<p>Specifies the Extensible Authentication Protocol Method for Microsoft Challenge Handshake Authentication Protocol (CHAP). This protocol enables extensible authentication for network access.</p> <p><a href="#">Click here to view this version of the [MS-CHAP] PDF.</a></p>

Specification	Description
<a href="#">[MS-CIFS]: Common Internet File System (CIFS) Protocol</a>	<p>Specifies the Common Internet File System (CIFS) Protocol, a cross-platform, transport-independent protocol that provides a mechanism for client systems to use file and print services made available by server systems over a network.</p> <p><a href="#">Click here to view this version of the [MS-CIFS] PDF.</a> ↗</p>
<a href="#">[MS-CMOM]: MSDTC Connection Manager: OleTx Management Protocol</a>	<p>Specifies the MSDTC Connection Manager: OleTx Management Protocol. This protocol enables the remote management of an OleTx Transaction Manager and its extensions.</p> <p><a href="#">Click here to view this version of the [MS-CMOM] PDF.</a> ↗</p>
<a href="#">[MS-CMP]: MSDTC Connection Manager: OleTx Multiplexing Protocol</a>	<p>Specifies the MSDTC Connection Manager Protocol: Connection Multiplexing Protocol, which enables partners to multiplex any number of two-way connections over the MSDTC Connection Manager: OleTx Transports Protocol session.</p> <p><a href="#">Click here to view this version of the [MS-CMP] PDF.</a> ↗</p>
<a href="#">[MS-CMPO]: MSDTC Connection Manager: OleTx Transports Protocol</a>	<p>Specifies the MSDTC Connection Manager: OleTx Transports Protocol, a peer-to-peer messaging protocol layered over a bidirectional pair of RPC connections.</p> <p><a href="#">Click here to view this version of the [MS-CMPO] PDF.</a> ↗</p>
<a href="#">[MS-CMRP]: Failover Cluster: Management API (ClusAPI) Protocol</a>	<p>Specifies the Failover Cluster: Management API (ClusAPI) Protocol, an RPC-based protocol that is used for remotely managing a cluster.</p> <p><a href="#">Click here to view this version of the [MS-CMRP] PDF.</a> ↗</p>
<a href="#">[MS-COM]: Component Object Model Plus (COM+) Protocol</a>	<p>Specifies the Component Object Model Plus (COM+) Protocol, which consists of a DCOM interface (and DCOM protocol extensions) that is used for adding transactions, implementing synchronization, managing multiple object class configurations, enforcing security, and providing additional functionality and attributes to DCOM-based distributed object applications.</p> <p><a href="#">Click here to view this version of the [MS-COM] PDF.</a> ↗</p>
<a href="#">[MS-COMA]: Component Object Model Plus (COM+) Remote Administration Protocol</a>	<p>Specifies the Component Object Model Plus (COM+) Remote Administration Protocol, which enables remote clients to register, import, remove, configure, control, and monitor components and conglomerations for an Object Request Broker (ORB).</p> <p><a href="#">Click here to view this version of the [MS-COMA] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-COMEV]: Component Object Model Plus (COM+) Event System Protocol</a>	<p>Specifies the Component Object Model Plus (COM+) Event System Protocol, which is a protocol that exposes DCOM interfaces for storing and managing configuration data for publishers of events and their respective subscribers on remote computers. This protocol also specifies how to get specific information about a publisher and its subscribers.</p> <p><a href="#">Click here to view this version of the [MS-COMEV] PDF.</a></p>
<a href="#">[MS-COMT]: Component Object Model Plus (COM+) Tracker Service Protocol</a>	<p>Specifies the Component Object Model Plus (COM+) Tracker Service Protocol, which enables clients to monitor running instances of components.</p> <p><a href="#">Click here to view this version of the [MS-COMT] PDF.</a></p>
<a href="#">[MS-CPSP]: Connection Point Services: Phonebook Data Structure</a>	<p>Specifies the Connection Point Services: Phonebook Data Structure. This structure describes a format for documenting POP entry information and a logical grouping of POPs based on their geographic location.</p> <p><a href="#">Click here to view this version of the [MS-CPSP] PDF.</a></p>
<a href="#">[MS-CRTD]: Certificate Templates Structure</a>	<p>Specifies the Certificate Templates Structure. This structure describes the syntax and interpretation of certificate templates, which forms the basis of certificate management for the Certificate Templates Protocol.</p> <p><a href="#">Click here to view this version of the [MS-CRTD] PDF.</a></p>
<a href="#">[MS-CSRA]: Certificate Services Remote Administration Protocol</a>	<p>Specifies the Certificate Services Remote Administration Protocol, which consists of a set of Distributed Component Object Model (DCOM) interfaces that enable administrative tools to configure the state and policy of a certification authority (CA) on a server.</p> <p><a href="#">Click here to view this version of the [MS-CSRA] PDF.</a></p>
<a href="#">[MS-CSSP]: Credential Security Support Provider (CredSSP) Protocol</a>	<p>Specifies the Credential Security Support Provider (CredSSP) Protocol, which enables an application to securely delegate a user's credentials from a client to a target server.</p> <p><a href="#">Click here to view this version of the [MS-CSSP] PDF.</a></p>
<a href="#">[MS-CSVP]: Failover Cluster: Setup and Validation Protocol (ClusPrep)</a>	<p>Specifies the Failover Cluster: Setup and Validation Protocol (ClusPrep), which remotely configures cluster nodes, cleans up cluster nodes, and validates that hardware and software settings are compatible with Failover Clustering.</p> <p><a href="#">Click here to view this version of the [MS-CSVP] PDF.</a></p>

Specification	Description
<a href="#">[MS-CTA]: Claims Transformation Algorithm</a>	<p>Specifies the Claims Transformation Algorithm (CTA), which consists of two components: a grammar describing a transformation rules language and an algorithm for transforming input claims into output claims. A claim is an assertion about a user identity in the form of a name-value tuple. Sets of claims are transformed from sending authority formats to receiving authority formats at authentication trust traversal boundaries.</p> <p><a href="#">Click here to view this version of the [MS-CTA] PDF.</a></p>
<a href="#">[MS-DCHT]: Desktop Chat Protocol</a>	<p>Specifies the Desktop Chat Protocol, which is the mechanism by which the Windows Chat application in Windows communicates information between remote users.</p> <p><a href="#">Click here to view this version of the [MS-DCHT] PDF.</a></p>
<a href="#">[MS-DCLB]: Desktop Clipboard Protocol</a>	<p>Specifies the Desktop Clipboard Protocol, which uses the Network Dynamic Data Exchange (NetDDE) Protocol to implement a distributed store for graphical user interface (GUI) objects for desktop cut-and-paste operations.</p> <p><a href="#">Click here to view this version of the [MS-DCLB] PDF.</a></p>
<a href="#">[MS-DCOM]: Distributed Component Object Model (DCOM) Remote Protocol</a>	<p>Specifies the Distributed Component Object Model (DCOM) Remote Protocol, which exposes application objects via remote procedure calls (RPCs) and consists of a set of extensions layered on the Microsoft Remote Procedure Call Extensions.</p> <p><a href="#">Click here to view this version of the [MS-DCOM] PDF.</a></p>
<a href="#">[MS-DFSC]: Distributed File System (DFS): Referral Protocol</a>	<p>Specifies the Distributed File System (DFS): Referral Protocol, which enables file system clients to resolve names from a namespace distributed across many servers and geographies into local names on specific file servers.</p> <p><a href="#">Click here to view this version of the [MS-DFSC] PDF.</a></p>
<a href="#">[MS-DFSNM]: Distributed File System (DFS): Namespace Management Protocol</a>	<p>Specifies the Distributed File System (DFS): Namespace Management Protocol, which provides an RPC interface for administering DFS configurations. The client is an application that issues method calls on the RPC interface to administer DFS. The server is a DFS service that implements support for this RPC interface for administering DFS.</p> <p><a href="#">Click here to view this version of the [MS-DFSNM] PDF.</a></p>
<a href="#">[MS-DFSRH]: DFS Replication Helper Protocol</a>	<p>Specifies the DFS Replication Helper Protocol, which is made up of a set of distributed component object model (DCOM) interfaces for configuring and monitoring DFS Replication Helper Protocols on a server.</p> <p><a href="#">Click here to view this version of the [MS-DFSRH] PDF.</a></p>

Specification	Description
<a href="#">[MS-DHA]: Device Health Attestation Protocol</a>	<p>Specifies the Device Health Attestation Service Protocol, which enables the assessment of the attested boot state of devices. The outcome of the health assessment is included in a signed health certificate which can then be evaluated by other services to determine whether a device is meeting enterprise corporate policy for device health, or by third party services to identify jailbroken devices that should not receive content or access to certain resources.</p> <p><a href="#">Click here to view this version of the [MS-DHA] PDF.</a></p>
<a href="#">[MS-DHCPE]: Dynamic Host Configuration Protocol (DHCP) Extensions</a>	<p>Specifies the Dynamic Host Configuration Protocol (DHCP), which describes the Microsoft specific vendor-class options included in the Microsoft implementation of DHCP.</p> <p><a href="#">Click here to view this version of the [MS-DHCPE] PDF.</a></p>
<a href="#">[MS-DHCPF]: DHCP Failover Protocol Extension</a>	<p>Specifies the DHCP Failover Protocol Extension, which extends the DHCP Failover Protocol by encrypting messages sent between the servers in a failover relationship and by providing client implementation options.</p> <p><a href="#">Click here to view this version of the [MS-DHCPF] PDF.</a></p>
<a href="#">[MS-DHCPM]: Microsoft Dynamic Host Configuration Protocol (DHCP) Server Management Protocol</a>	<p>Specifies the Microsoft Dynamic Host Configuration Protocol (DHCP) Server Management Protocol, which defines the RPC interfaces that provide methods for remotely accessing and administering the DHCP server. This protocol is a client and server protocol based on RPC that is used in the configuration, management, and monitoring of a DHCP server.</p> <p><a href="#">Click here to view this version of the [MS-DHCPM] PDF.</a></p>
<a href="#">[MS-DHCPN]: Dynamic Host Configuration Protocol (DHCP) Extensions for Network Access Protection (NAP)</a>	<p>Specifies the Dynamic Host Configuration Protocol (DHCP) Extensions for Network Access Protection (NAP), which is designed to reduce the administrative burden and complexity of configuring hosts on a TCP/IP-based network, such as a private intranet, and is one enforcement method supported by Network Access Protection (NAP).</p> <p><a href="#">Click here to view this version of the [MS-DHCPN] PDF.</a></p>
<a href="#">[MS-DLNHND]: Digital Living Network Alliance (DLNA) Networked Device Interoperability Guidelines: Microsoft Extensions</a>	<p>Specifies Digital Living Network Alliance (DLNA) Home Networked Device Interoperability Guidelines: Microsoft Extensions. The DLNA Guidelines define protocol extensions to protocols related to streaming of content.</p> <p><a href="#">Click here to view this version of the [MS-DLNHND] PDF.</a></p>

Specification	Description
<a href="#">[MS-DLTC]: Distributed Link Tracking Central Store Protocol</a>	<p>Specifies the Distributed Link Tracking Central Store Protocol, which defines how the Active Directory objects are defined, updated, and interpreted. [MS-DLTC] works with the Distributed Link Tracking (DLT) Workstation Protocol and the DLT Central Manager Protocol ([MS-DLTM]), the two other protocols that make up Distributed Link Tracking.</p> <p><a href="#">Click here to view this version of the [MS-DLTC] PDF.</a></p>
<a href="#">[MS-DLTM]: Distributed Link Tracking: Central Manager Protocol</a>	<p>Specifies the Distributed Link Tracking: Central Manager Protocol, which works with the Distributed Link Tracking (DLT) Workstation Protocol to discover the new location of a file that has moved. DLT can determine whether the file has moved on a mass-storage device, within a computer, or between computers in a network. The DLT Central Manager Protocol keeps track of file and volume moves and other relevant information from participating computers in order to provide this information in response to workstation queries.</p> <p><a href="#">Click here to view this version of the [MS-DLTM] PDF.</a></p>
<a href="#">[MS-DLTW]: Distributed Link Tracking: Workstation Protocol</a>	<p>Specifies the Distributed Link Tracking: Workstation Protocol, which works with the Distributed Link Tracking (DLT) Central Manager Protocol to discover the new location of a file that has moved. DLT can determine whether the file has moved on a mass-storage device, within a computer, or between computers in a network.</p> <p><a href="#">Click here to view this version of the [MS-DLTW] PDF.</a></p>
<a href="#">[MS-DMCT]: Device Media Control Protocol</a>	<p>Specifies the Device Media Control Protocol, which uses the Device Services Lightweight Remoting Protocol [MS-DSLR] to enable a computer to control media playback in an active device session.</p> <p><a href="#">Click here to view this version of the [MS-DMCT] PDF.</a></p>
<a href="#">[MS-DMRP]: Disk Management Remote Protocol</a>	<p>Specifies the Disk Management Remote Protocol, a set of Distributed Component Object Model (DCOM) interfaces that manages storage objects on a machine.</p> <p><a href="#">Click here to view this version of the [MS-DMRP] PDF.</a></p>
<a href="#">[MS-DNSP]: Domain Name Service (DNS) Server Management Protocol</a>	<p>Specifies the Domain Name Service (DNS) Server Management Protocol, which defines the RPC interfaces that provide methods for remotely accessing and administering a DNS server. It is a client and server protocol based on RPC that is used in the configuration, management, and monitoring of a DNS server.</p> <p><a href="#">Click here to view this version of the [MS-DNSP] PDF.</a></p>

Specification	Description
<a href="#">[MS-DPDX]: DirectPlay DXDiag Usage Protocol</a>	<p>Specifies the DirectPlay DXDiag Usage Protocol, intended for peer-to-peer network video gaming and used by the DXDiag application.</p> <p><a href="#">Click here to view this version of the [MS-DPDX] PDF.</a></p>
<a href="#">[MS-DPSP]: Digest Protocol Extensions</a>	<p>Specifies the Digest Protocol Extensions, which describes the variations in the Windows implementation of the Digest Authentication protocol from the standard, as specified in [RFC2617].</p> <p><a href="#">Click here to view this version of the [MS-DPSP] PDF.</a></p>
<a href="#">[MS-DPWSRP]: Devices Profile for Web Services (DPWS): Shared Resource Publishing Data Structure</a>	<p>Specifies the Shell Publishing data structure. This data structure is used by the HomeGroup Protocol to advertise shared files and folders in a HomeGroup peer-to-peer network environment.</p> <p><a href="#">Click here to view this version of the [MS-DPWSRP] PDF.</a></p>
<a href="#">[MS-DPWSSN]: Devices Profile for Web Services (DPWS): Size Negotiation Extension</a>	<p>Specifies the Devices Profile for Web Services (DPWS): Size Negotiation Extension. This is an extension to the Devices Profile for Web Services (DPWS) and enables the negotiation of message sizes between a client and a service for a specific message transaction.</p> <p><a href="#">Click here to view this version of the [MS-DPWSSN] PDF.</a></p>
<a href="#">[MS-DRM]: Digital Rights Management License Protocol</a>	<p>Specifies the Digital Rights Management License Protocol, which provides secure distribution, promotion, and sale of digital media content.</p> <p><a href="#">Click here to view this version of the [MS-DRM] PDF.</a></p>
<a href="#">[MS-DRMCD]: Windows Media Digital Rights Management (WMDRM): MTP Command Extension</a>	<p>Specifies the Media Transfer Protocol (MTP): WMDRM Portable Device Extensions, which support digital rights management for portable consumer electronic devices. These protocol extensions can be used to enable consumers to experience audio and/or video on portable devices, while protecting the rights of the content owner.</p> <p><a href="#">Click here to view this version of the [MS-DRMCD] PDF.</a></p>
<a href="#">[MS-DRMND]: Windows Media Digital Rights Management (WMDRM): Network Devices Protocol</a>	<p>Specifies the Windows Media Digital Rights Management (WMDRM): Network Devices Protocol. This protocol enables consumers to experience multimedia content on multiple devices in the home, while protecting the rights of the content owner.</p> <p><a href="#">Click here to view this version of the [MS-DRMND] PDF.</a></p>

Specification	Description
<a href="#">[MS-DRMRI]: Windows Media Digital Rights Management for Network Devices (WMDRM-ND): Registrar Initiation Protocol</a>	<p>Specifies the Windows Media Digital Rights Management for Network Devices (WMDRM-ND): Registrar Initiation Protocol, a set of services provided by a host (for example, a personal computer) and a client (for example, an extender device) that allows a WMDRM-ND registration and authentication process to be remotely initiated and completed between the host and client. This allows DRM-protected contents stored on the host to be shared securely with the client.</p> <p><a href="#">Click here to view this version of the [MS-DRMRI] PDF.</a></p>
<a href="#">[MS-DRSR]: Directory Replication Service (DRS) Remote Protocol</a>	<p>Specifies the Directory Replication Service (DRS) Remote Protocol, an RPC protocol for replication and management of data in Active Directory.</p> <p><a href="#">Click here to view this version of the [MS-DRSR] PDF.</a></p>
<a href="#">[MS-DSCPM]: Desired State Configuration Pull Model Protocol</a>	<p>Specifies the Desired State Configuration Pull Model Protocol, which is used to get a client's configuration and modules from the server and to report the client's status back to the server. The protocol depends on HTTP for the transfer of all protocol messages. With the Desired State Configuration Pull Model Protocol, binary data flows from the server to the client.</p> <p><a href="#">Click here to view this version of the [MS-DSCPM] PDF.</a></p>
<a href="#">[MS-DSLR]: Device Services Lightweight Remoting Protocol</a>	<p>Specifies the Device Services Lightweight Remoting Protocol, which enables remoting of services (objects, function calls, events, and so on) over a reliable point-to-point channel.</p> <p><a href="#">Click here to view this version of the [MS-DSLR] PDF.</a></p>
<a href="#">[MS-DSML]: Directory Services Markup Language (DSML) 2.0 Protocol Extensions</a>	<p>Specifies the Directory Services Markup Language (DSML) 2.0 Protocol Extensions. The SOAP session extensions (SSE) make it possible to maintain state information across multiple request/response operations.</p> <p><a href="#">Click here to view this version of the [MS-DSML] PDF.</a></p>
<a href="#">[MS-DSMN]: Device Session Monitoring Protocol</a>	<p>Specifies the Device Session Monitoring Protocol, which enables a client device to monitor the status of the host in a remote session. DSMN is built on the Device Services Lightweight Remoting Protocol [MS-DSLR].</p> <p><a href="#">Click here to view this version of the [MS-DSMN] PDF.</a></p>
<a href="#">[MS-DSPA]: Device Session Property Access Protocol</a>	<p>Specifies the Device Session Property Access Protocol, which enables a computer to exchange name-value pairs with a device in an active device session. The Device Session Property Access Protocol uses the Device Services Lightweight Remoting Protocol [MS-DSLR] to enable the exchange.</p> <p><a href="#">Click here to view this version of the [MS-DSPA] PDF.</a></p>

Specification	Description
<a href="#">[MS-DSSP]: Directory Services Setup Remote Protocol</a>	<p>Specifies the Directory Services Setup Remote Protocol, which exposes an RPC interface that a client can call to obtain domain-related computer state and configuration information.</p> <p><a href="#">Click here to view this version of the [MS-DSSP] PDF.</a> ↗</p>
<a href="#">[MS-DTAG]: Device Trust Agreement Protocol</a>	<p>Specifies the Device Trust Agreement Protocol, which enables two UPnP endpoints to securely exchange certificates over an unsecure network and to establish a trust relationship by means of a simple, one-time shared secret.</p> <p><a href="#">Click here to view this version of the [MS-DTAG] PDF.</a> ↗</p>
<a href="#">[MS-DTCLU]: MSDTC Connection Manager: OleTx Transaction Protocol Logical Unit Mainframe Extension</a>	<p>Specifies the MSDTC Connection Manager: OleTx Transaction Protocol Logical Unit Mainframe Extension, which provides concrete mechanisms for associating an Atomic Transaction and an LU type 6.2 Logical Unit of Work.</p> <p><a href="#">Click here to view this version of the [MS-DTCLU] PDF.</a> ↗</p>
<a href="#">[MS-DTCM]: MSDTC Connection Manager: OleTx Transaction Internet Protocol</a>	<p>Specifies the MSDTC Connection Manager: OleTx Transaction Internet Protocol, which extends the OleTx protocol (see [MS-DTCO]) to enable its interoperability with the open-standard Transaction Internet Protocol (TIP).</p> <p><a href="#">Click here to view this version of the [MS-DTCM] PDF.</a> ↗</p>
<a href="#">[MS-DTCO]: MSDTC Connection Manager: OleTx Transaction Protocol</a>	<p>Specifies the MSDTC Connection Manager: OleTx Transaction Protocol, which provides concrete mechanisms for beginning, propagating, and completing atomic transactions. This protocol also provides mechanisms for coordinating agreement on a single atomic outcome for each transaction, and for reliably distributing that outcome to all participants in the transaction.</p> <p><a href="#">Click here to view this version of the [MS-DTCO] PDF.</a> ↗</p>
<a href="#">[MS-DVRD]: Device Registration Discovery Protocol</a>	<p>Specifies the Device Registration Discovery Protocol, which is used to discover information about servers that can register corporate-owned and personal devices with a corporate network.</p> <p><a href="#">Click here to view this version of the [MS-DVRD] PDF.</a> ↗</p>
<a href="#">[MS-DVRE]: Device Registration Enrollment Protocol</a>	<p>Specifies the Device Registration Enrollment Protocol, which is used to register corporate-owned and personal devices with a corporate network.</p> <p><a href="#">Click here to view this version of the [MS-DVRE] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-DVRJ]: Device Registration Join Protocol</a>	<p>Specifies the Device Registration Join Protocol, which establishes a device identity between the physical device and a directory service. The identity is used at the system level to identify the device only.</p> <p><a href="#">Click here to view this version of the [MS-DVRJ] PDF.</a></p>
<a href="#">[MS-ECS]: Enterprise Client Synchronization Protocol</a>	<p>Specifies the Enterprise Client Sync protocol, which enables devices (such as tablets, PCs, or laptops) to synchronize files to and from a file server in a REST-based manner.</p> <p><a href="#">Click here to view this version of the [MS-ECS] PDF.</a></p>
<a href="#">[MS-EERR]: ExtendedError Remote Data Structure</a>	<p>Specifies the ExtendedError Remote Data Structure, which encodes extended error information. This data structure assumes that the reader has familiarity with the concepts and the requirements that are detailed in [MS-RPCE] and [C706].</p> <p><a href="#">Click here to view this version of the [MS-EERR] PDF.</a></p>
<a href="#">[MS-EFSR]: Encrypting File System Remote (EFSRPC) Protocol</a>	<p>Specifies the Encrypting File System Remote (EFSRPC) Protocol, which performs maintenance and management operations on encrypted data that is stored remotely and accessed over a network.</p> <p><a href="#">Click here to view this version of the [MS-EFSR] PDF.</a></p>
<a href="#">[MS-EMF]: Enhanced Metafile Format</a>	<p>Specifies the Enhanced Metafile Format (EMF) structure, which can store a picture in device-independent form.</p> <p><a href="#">Click here to view this version of the [MS-EMF] PDF.</a></p>
<a href="#">[MS-EMFPLUS]: Enhanced Metafile Format Plus Extensions</a>	<p>Specifies the Enhanced Metafile Format Plus Extensions, which defines a device-independent structure that encapsulates graphics commands and objects for storage or for sending to devices, such as displays and printers that support the drawing of images, graphics, and text.</p> <p><a href="#">Click here to view this version of the [MS-EMFPLUS] PDF.</a></p>
<a href="#">[MS-EMFSPOOL]: Enhanced Metafile Spool Format</a>	<p>Specifies the Enhanced Metafile Spool Format. This structure specifies a metafile format that can store a print job in portable form.</p> <p><a href="#">Click here to view this version of the [MS-EMFSPOOL] PDF.</a></p>
<a href="#">[MS-EVEN]: EventLog Remoting Protocol</a>	<p>Specifies the EventLog Remoting Protocol, which exposes the RPC methods for reading events in both live and backup event logs on remote computers.</p> <p><a href="#">Click here to view this version of the [MS-EVEN] PDF.</a></p>

Specification	Description
<a href="#">[MS-EVEN6]: EventLog Remoting Protocol Version 6.0</a>	<p>Specifies the EventLog Remoting Protocol Version 6.0 protocol, which exposes RPC methods for reading events in both live and backup event logs on remote computers. This protocol was originally made available for Windows Vista.</p> <p><a href="#">Click here to view this version of the [MS-EVEN6] PDF.</a> ↗</p>
<a href="#">[MS-FASP]: Firewall and Advanced Security Protocol</a>	<p>Specifies the Firewall and Advanced Security Protocol. The protocol manages firewall and advanced security components on remote computers.</p> <p><a href="#">Click here to view this version of the [MS-FASP] PDF.</a> ↗</p>
<a href="#">[MS-FAX]: Fax Server and Client Remote Protocol</a>	<p>Specifies the Fax Server and Client Remote Protocol. It is an RPC-based, client-server protocol, and is used to send faxes and to manage the fax server and its queues.</p> <p><a href="#">Click here to view this version of the [MS-FAX] PDF.</a> ↗</p>
<a href="#">[MS-FCIADS]: File Classification Infrastructure Alternate Data Stream (ADS) File Format</a>	<p>Specifies the File Classification Infrastructure Alternate Data Stream (ADS) File Format, which consists of structures for persisting file metadata information into NTFS alternate data streams.</p> <p><a href="#">Click here to view this version of the [MS-FCIADS] PDF.</a> ↗</p>
<a href="#">[MS-FRS1]: File Replication Service Protocol</a>	<p>Specifies the File Replication Service Protocol, which is a replication protocol that is used to replicate files and folders across one or more members in an Active Directory domain. It works to keep copies of a file system tree up to date on all members of a replication group, while allowing any member of the group to change the contents at any time.</p> <p><a href="#">Click here to view this version of the [MS-FRS1] PDF.</a> ↗</p>
<a href="#">[MS-FRS2]: Distributed File System Replication Protocol</a>	<p>Specifies the SD Microsoft Distributed File System Replication Protocol, which defines an RPC interface that replicates files between servers and enables the creation of multimaster optimistic file replication systems.</p> <p><a href="#">Click here to view this version of the [MS-FRS2] PDF.</a> ↗</p>
<a href="#">[MS-FSA]: File System Algorithms</a>	<p>Specifies File System Algorithms in terms of an abstract model for how an object store can be implemented to support the Server Message Block (SMB) Version 1.0 Protocol [MS-SMB] and the Server Message Block (SMB) Version 2.0 Protocol [MS-SMB2].</p> <p><a href="#">Click here to view this version of the [MS-FSA] PDF.</a> ↗</p>

Specification	Description
[MS-FSCC]: File System Control Codes	<p>Specifies the File System Control Codes that define the network format of native Windows structures that may be used within other protocols.</p> <p><a href="#">Click here to view this version of the [MS-FSCC] PDF.</a> ↗</p>
[MS-FSRM]: File Server Resource Manager Protocol	<p>Specifies the File Server Resource Manager Protocol, which implements a set of a Distributed Component Object Model (DCOM) interfaces for managing the configuration of directory quotas, file screens, and storage report jobs on a machine.</p> <p><a href="#">Click here to view this version of the [MS-FSRM] PDF.</a> ↗</p>
[MS-FSRVP]: File Server Remote VSS Protocol	<p>Specifies the File Server Remote VSS Protocol, an RPC-based protocol used for creating shadow copies of file shares on a remote computer, and for facilitating backup applications in performing application-consistent backup and restore of data on SMB2 shares.</p> <p><a href="#">Click here to view this version of the [MS-FSRVP] PDF.</a> ↗</p>
[MS-FSVCA]: File Set Version Comparison Algorithms	<p>Specifies the File Set Version Comparison Algorithms, which is used by the Enterprise Client Synchronization Protocol to build and serialize a compact representation of version state across a data set consisting of files and directories.</p> <p><a href="#">Click here to view this version of the [MS-FSVCA] PDF.</a> ↗</p>
[MS-FTPS]: File Transfer Protocol over Secure Sockets Layer (FTPS)	<p>Specifies an extension to the File Transfer Protocol over TLS (FTPS). This extends FTPS with a feature known as Implicit SSL and introduces the AUTH SSL message to allow interoperability with legacy FTP clients.</p> <p><a href="#">Click here to view this version of the [MS-FTPS] PDF.</a> ↗</p>
[MS-GKDI]: Group Key Distribution Protocol	<p>Specifies the Group Key Distribution Protocol, which enables clients to obtain cryptographic keys associated with Active Directory security principals.</p> <p><a href="#">Click here to view this version of the [MS-GKDI] PDF.</a> ↗</p>
[MS-GPAC]: Group Policy: Audit Configuration Extension	<p>Specifies the Group Policy: Audit Configuration Extension, which provides a mechanism for an administrator to control audit policies on clients.</p> <p><a href="#">Click here to view this version of the [MS-GPAC] PDF.</a> ↗</p>
[MS-GPCAP]: Group Policy: Central Access Policies Protocol Extension	<p>Specifies the Group Policy: Central Access Policies Extension, which provides the means of configuring central access policies that are applied to Group Policy client computer resources for authorization purposes.</p> <p><a href="#">Click here to view this version of the [MS-GPCAP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-GPDPC]: Group Policy: Deployed Printer Connections Extension</a>	<p>Specifies the Group Policy: Deployed Printer Connections Extension, which supports the use of preconfigured collections of shared printer connections.</p> <p><a href="#">Click here to view this version of the [MS-GPDPC] PDF.</a></p>
<a href="#">[MS-GPEF]: Group Policy: Encrypting File System Extension</a>	<p>Specifies the Group Policy: Encrypting File System Extension, which uses the Microsoft Group Policy Protocol to enable remote administrative configuration of the Encrypting File System.</p> <p><a href="#">Click here to view this version of the [MS-GPEF] PDF.</a></p>
<a href="#">[MS-GPFSAS]: Group Policy: Firewall and Advanced Security Data Structure</a>	<p>Specifies The Group Policy: Firewall and Advanced Security data structure extension, which provides a mechanism for an administrator to control the Firewall and Advanced Security behavior of the client through group policy by using the Group Policy: Registry Extension Encoding protocol [MS-GPREG].</p> <p><a href="#">Click here to view this version of the [MS-GPFSAS] PDF.</a></p>
<a href="#">[MS-GPFR]: Group Policy: Folder Redirection Protocol Extension</a>	<p>Specifies the Group Policy: Folder Redirection Protocol Extension, which provides a mechanism to relocate specific user folders to server disk volumes. The protocol extension describes how file system access requests to a user's folders are automatically redirected to a newly created folder for each user.</p> <p><a href="#">Click here to view this version of the [MS-GPFR] PDF.</a></p>
<a href="#">[MS-GPIE]: Group Policy: Internet Explorer Maintenance Extension</a>	<p>Specifies the Group Policy: Internet Explorer Maintenance Extension, which enables administrators to apply custom settings to the Internet Explorer configuration on one or more computers to enforce Internet-related security standards and provide a common browser interface within the organization.</p> <p><a href="#">Click here to view this version of the [MS-GPIE] PDF.</a></p>
<a href="#">[MS-GPIPSEC]: Group Policy: IP Security (IPsec) Protocol Extension</a>	<p>Specifies the IP Security (IPSec) Protocol Extension to the Group Policy: Core Protocol. This extension enables administrators to arbitrarily instruct large groups of client machines to configure their local IPsec/IKE components to provide basic IP traffic filtering, IP data integrity, and (optionally) IP data encryption.</p> <p><a href="#">Click here to view this version of the [MS-GPIPSEC] PDF.</a></p>
<a href="#">[MS-GPNAP]: Group Policy: Network Access Protection (NAP) Extension</a>	<p>Specifies the Group Policy: Network Access Protection (NAP) Extension, used for controlling access to network resources. This extension enables network administrators to grant or restrict access to network resources based on client computer identity and compliance with corporate governance policy.</p> <p><a href="#">Click here to view this version of the [MS-GPNAP] PDF.</a></p>

Specification	Description
<a href="#">[MS-GPNRPT]: Group Policy: Name Resolution Policy Table (NRPT) Data Extension</a>	<p>Specifies the Name Resolution Policy Table (NRPT) Group Policy Data Extension, an extension to Group Policy: Registry Extension Encoding [MS-GPREG]. The NRPT Group Policy Data Extension provides a mechanism for an administrator to control any Name Resolution Policy behavior on a client by using group policy-based settings.</p> <p><a href="#">Click here to view this version of the [MS-GPNRPT] PDF.</a> ↗</p>
<a href="#">[MS-GPOL]: Group Policy: Core Protocol</a>	<p>Specifies the Group Policy: Core Protocol, which enables clients to discover and retrieve policy settings that administrators of a domain create.</p> <p><a href="#">Click here to view this version of the [MS-GPOL] PDF.</a> ↗</p>
<a href="#">[MS-GPPREF]: Group Policy: Preferences Extension Data Structure</a>	<p>Specifies the Group Policy: Preferences Extension. This extension to the Group Policy: Core Protocol provides a mechanism to manage and deploy policy preferences.</p> <p><a href="#">Click here to view this version of the [MS-GPPREF] PDF.</a> ↗</p>
<a href="#">[MS-GPREG]: Group Policy: Registry Extension Encoding</a>	<p>Specifies the Group Policy: Registry Extension Encoding, an extension to the Group Policy: Core Protocol. This mechanism enables an administrator to control any behavior on a client that depends on registry-based settings.</p> <p><a href="#">Click here to view this version of the [MS-GPREG] PDF.</a> ↗</p>
<a href="#">[MS-GPSB]: Group Policy: Security Protocol Extension</a>	<p>Specifies the Group Policy: Security Protocol Extension, which is an extension to the Group Policy: Core Protocol. This extension enables security policies to be distributed to multiple client systems, so these systems can enact the policies in accordance with the intentions of the administrator.</p> <p><a href="#">Click here to view this version of the [MS-GPSB] PDF.</a> ↗</p>
<a href="#">[MS-GPSCR]: Group Policy: Scripts Extension Encoding</a>	<p>Specifies the Group Policy: Scripts Extension Encoding, an extension to the Group Policy: Core Protocol that provides a mechanism for an administrator to instruct an arbitrarily large group of clients to execute administrator-specified code at computer startup, computer shutdown, user logon, and user logoff.</p> <p><a href="#">Click here to view this version of the [MS-GPSCR] PDF.</a> ↗</p>
<a href="#">[MS-GPSI]: Group Policy: Software Installation Protocol Extension</a>	<p>Specifies the Group Policy: Software Installation Protocol Extension, which enables an administrator to install and remove software applications on client computers.</p> <p><a href="#">Click here to view this version of the [MS-GPSI] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-GPWL]: Group Policy: Wireless/Wired Protocol Extension</a>	<p>Specifies the Group Policy: Wireless/Wired Protocol Extension, an extension to the Group Policy: Core Protocol that specifies the behaviors of the Wireless/Wired Group Policy administrative-side and client-side plug-in extensions.</p> <p><a href="#">Click here to view this version of the [MS-GPWL] PDF.</a> ↗</p>
<a href="#">[MS-GSSA]: Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG) Protocol Extension</a>	<p>Specifies the Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG) Protocol Extension, which identifies one possible extension to TSIG based on the Generic Security Service Application Program Interface (GSS-API).</p> <p><a href="#">Click here to view this version of the [MS-GSSA] PDF.</a> ↗</p>
<a href="#">[MS-H245]: H.245 Protocol: Microsoft Extensions</a>	<p>Specifies the H.245 Protocol: Microsoft Extensions, which describes Microsoft extensions for the H.323 protocol.</p> <p><a href="#">Click here to view this version of the [MS-H245] PDF.</a> ↗</p>
<a href="#">[MS-H26XPF]: Real-Time Transport Protocol (RTP/RTCP): H.261 and H.263 Video Streams Extensions</a>	<p>Specifies the Real-Time Transport Protocol (RTP/RTCP): H.261 and H.263 Video Streams Extensions, which are used to transmit and receive H.261 or H.263 video streams in a two-party, peer-to-peer call.</p> <p><a href="#">Click here to view this version of the [MS-H26XPF] PDF.</a> ↗</p>
<a href="#">[MS-HCEP]: Health Certificate Enrollment Protocol</a>	<p>Specifies the Health Certificate Enrollment Protocol, which enables a network endpoint to obtain digital certificates.</p> <p><a href="#">Click here to view this version of the [MS-HCEP] PDF.</a> ↗</p>
<a href="#">[MS-HGRP]: HomeGroup Protocol</a>	<p>Specifies the HomeGroup Protocol, which is used to create a trust relationship that facilitates the advertising and publishing of content between machines via a peer-to-peer (P2P) infrastructure.</p> <p><a href="#">Click here to view this version of the [MS-HGRP] PDF.</a> ↗</p>
<a href="#">[MS-HGSA]: Host Guardian Service: Attestation Protocol</a>	<p>Specifies the Host Guardian Services Attestation (HGSA) protocol, one of two services that comprise the Host Guardian Service. Host Guardian Service is a server role that provides security assurance for Shielded Virtual Machines (VMs) by ensuring that Shielded VMs can be run only on known and trusted fabric hosts that have a legitimate configuration. The other component service, the Key Protection Service, is specified in the [MS-KPS] protocol document.</p> <p><a href="#">Click here to view this version of the [MS-HGSA] PDF.</a> ↗</p>

Specification	Description
[MS-HNDS]: Host Name Data Structure Extension	<p>Specifies the Host Name Data Structure Extension, which defines the allowable host names that may be assigned to a computer.</p> <p><a href="#">Click here to view this version of the [MS-HNDS] PDF.</a> ↗</p>
[MS-HRL]: Hyper-V Replica Log (HRL) File Format	<p>Specifies the Hyper-V Replica Log (HRL) File Format. Hyper-V Replica log files, required for tracking changes that have been made to the primary server, are created as part of failover replication. They are transported to the recovery server and parsed; updates are then applied to the recovery server.</p> <p><a href="#">Click here to view this version of the [MS-HRL] PDF.</a> ↗</p>
[MS-HTTP2E]: Hypertext Transfer Protocol Version 2 (HTTP/2) Extension	<p>Specifies a profile of and an extension to the Hypertext Transfer Protocol (HTTP) version 2, which is defined by [RFC7540].</p> <p><a href="#">Click here to view this version of the [MS-HTTP2E] PDF.</a> ↗</p>
[MS-HTTPE]: Hypertext Transfer Protocol (HTTP) Extensions	<p>Specifies the Hypertext Transfer Protocol (HTTP) Extensions, which extend the HyperText Transfer Protocol (HTTP) and deal with internationalization of host names and query strings.</p> <p><a href="#">Click here to view this version of the [MS-HTTPE] PDF.</a> ↗</p>
[MS-HVRS]: Hyper-V Remote Storage Profile	<p>Specifies information regarding the implementation for hosting Hyper-V virtual machine files on Server Message Block (SMB) Version 3 shares.</p> <p><a href="#">Click here to view this version of the [MS-HVRS] PDF.</a> ↗</p>
[MS-ICPR]: ICertPassage Remote Protocol	<p>Specifies the ICertPassage Remote Protocol, a subset of the Windows Client Certificate Enrollment Protocol, as specified in [MS-WCCE]. This protocol only enables the client to enroll certificates, whereas [MS-WCCE] provides enrollment and additional functionality.</p> <p><a href="#">Click here to view this version of the [MS-ICPR] PDF.</a> ↗</p>
[MS-IISS]: Internet Information Services (IIS) ServiceControl Protocol	<p>Specifies the Internet Information Services (IIS) ServiceControl Protocol, a client-to-server protocol that enables remote control of Internet services as a single unit.</p> <p><a href="#">Click here to view this version of the [MS-IISS] PDF.</a> ↗</p>
[MS-IKEE]: Internet Key Exchange Protocol Extensions	<p>Specifies the Internet Key Exchange (IKE) Protocol Extensions, which describe the extensions specified in [RFC2409].</p> <p><a href="#">Click here to view this version of the [MS-IKEE] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-IMSA]: Internet Information Services (IIS) IMSAdminBaseW Remote Protocol</a>	<p>Specifies the Internet Information Services (IIS) IMSAdminBaseW Remote Protocol, which defines interfaces that provide Unicode-compliant methods for remotely accessing and administering the IIS metabase associated with an application that manages IIS configuration, such as the IIS snap-in for Microsoft Management Console (MMC).</p> <p><a href="#">Click here to view this version of the [MS-IMSA] PDF.</a> ↗</p>
<a href="#">[MS-IOI]: IManagedObject Interface Protocol</a>	<p>Specifies the IManagedObject Interface Protocol. The IManagedObject interface is a COM interface used by the common language runtime (CLR) to identify managed objects (objects created by the CLR) that are exported for interoperability with the Component Object Model (COM).</p> <p><a href="#">Click here to view this version of the [MS-IOI] PDF.</a> ↗</p>
<a href="#">[MS-IPAMM]: IP Address Management (IPAM) Management Protocol</a>	<p>Specifies the IP Address Management (IPAM) Management Protocol. This protocol is used to remotely retrieve and manage the data in the IPAM data store. The IPAM data store consists of the data pertaining to address space management, which includes the configuration data available with the DHCP and DNS server instances in the network.</p> <p><a href="#">Click here to view this version of the [MS-IPAMM] PDF.</a> ↗</p>
<a href="#">[MS-IPAMM2]: IP Address Management (IPAM) Management Protocol Version 2</a>	<p>Specifies the IP Address Management (IPAM) Management Protocol. This protocol is used to remotely retrieve and manage the data in the IPAM data store. The IPAM data store consists of the data pertaining to the address space management, which includes the configuration data available with the DHCP and DNS server instances in the network.</p> <p><a href="#">Click here to view this version of the [MS-IPAMM2] PDF.</a> ↗</p>
<a href="#">[MS-IPHTTPS]: IP over HTTPS (IP-HTTPS) Tunneling Protocol</a>	<p>Specifies the IP over HTTPS (IP-HTTPS) Tunneling Protocol, a mechanism to transport IPv6 packets on an HTTPS connection.</p> <p><a href="#">Click here to view this version of the [MS-IPHTTPS] PDF.</a> ↗</p>
<a href="#">[MS-IRDA]: IrDA Object Exchange (OBEX) Protocol Profile</a>	<p>Specifies the IrDA Object Exchange (OBEX) Protocol Profile, which clarifies the implementation details of [IROBEX] where necessary and clarifies which portions of [IROBEX] are not implemented.</p> <p><a href="#">Click here to view this version of the [MS-IRDA] PDF.</a> ↗</p>
<a href="#">[MS-IRP]: Internet Information Services (IIS) Inetinfo Remote Protocol</a>	<p>Specifies the Internet Information Services (IIS) Inetinfo Remote Protocol, an RPC-based client/server protocol that is used for managing Internet protocol servers such as those hosted by IIS.</p> <p><a href="#">Click here to view this version of the [MS-IRP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-KILE]: Kerberos Protocol Extensions</a>	<p>Specifies the Microsoft implementation of the Kerberos Protocol Extensions, as specified in [RFC4120], by specifying any Windows behaviors that differ from the Kerberos Protocol, in addition to Windows extensions for interactive logon and the inclusion of authorization information expressed as group memberships and related information.</p> <p><a href="#">Click here to view this version of the [MS-KILE] PDF.</a> ↗</p>
<a href="#">[MS-KKDCP]: Kerberos Key Distribution Center (KDC) Proxy Protocol</a>	<p>Specifies the Kerberos Key Distribution Center (KDC) Proxy Protocol, which provides a mechanism for a client to use a KKDCP server to change passwords and securely obtain Kerberos service tickets from a Kerberos V5 server.</p> <p><a href="#">Click here to view this version of the [MS-KKDCP] PDF.</a> ↗</p>
<a href="#">[MS-KPP]: Key Provisioning Protocol</a>	<p>Specifies the Key Provisioning Protocol, which defines a mechanism for a client to register a set of cryptographic keys on a user and device pair.</p> <p><a href="#">Click here to view this version of the [MS-KPP] PDF.</a> ↗</p>
<a href="#">[MS-KPS]: Key Protection Service Protocol</a>	<p>Specifies the Key Protection Service protocol, one of two services that comprise the Host Guardian Service. Host Guardian Service is a server role that provides security assurance for Shielded Virtual Machines (VMs) by ensuring that Shielded VMs can be run only on known and trusted fabric hosts that have a legitimate configuration. The other component service, the Attestation Service, is specified in the [MS-HGSA] protocol document.</p> <p><a href="#">Click here to view this version of the [MS-KPS] PDF.</a> ↗</p>
<a href="#">[MS-L2TPIE]: Layer 2 Tunneling Protocol (L2TP) IPsec Extensions</a>	<p>Specifies the Layer 2 Tunneling Protocol (L2TP) IPsec Extensions, which allows IP, IPX, or NetBEUI traffic to be encrypted and then sent over any medium that supports point-to-point (PPP) (Point to Point Protocol [RFC1661]) datagram delivery, such as IP, X.25, Frame Relay, or ATM.</p> <p><a href="#">Click here to view this version of the [MS-L2TPIE] PDF.</a> ↗</p>
<a href="#">[MS-LLMNR]: Link Local Multicast Name Resolution (LLMNR) Profile</a>	<p>Specifies the Link Local Multicast Name Resolution (LLMNR) Profile, which describes the differences between this profile and the one defined in [RFC4795].</p> <p><a href="#">Click here to view this version of the [MS-LLMNR] PDF.</a> ↗</p>
<a href="#">[MS-LLTD]: Link Layer Topology Discovery (LLTD) Protocol</a>	<p>Specifies the Link Layer Topology Discovery (LLTD) Protocol, which an application or a higher-layer protocol can use to facilitate discovery of link-layer topology and diagnose various problems associated with a network's signal strength and bandwidth.</p> <p><a href="#">Click here to view this version of the [MS-LLTD] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-LREC]: Live Remote Event Capture (LREC) Protocol</a>	<p>Specifies the Live Remote Event Capture (LREC) Protocol, which enables a management station to monitor events on a target system across a network. The protocol supports various monitoring scenarios, such as a ""first line of defense"" for troubleshooting, where the remote system does not support the ability to log events locally.</p> <p><a href="#">Click here to view this version of the [MS-LREC] PDF.</a></p>
<a href="#">[MS-LSAD]: Local Security Authority (Domain Policy) Remote Protocol</a>	<p>Specifies the Local Security Authority (Domain Policy) Remote Protocol. This protocol provides an RPC interface used for providing remote management for policy settings related to account objects, secret objects, trusted domain objects (TDOs), and other security-related policy settings.</p> <p><a href="#">Click here to view this version of the [MS-LSAD] PDF.</a></p>
<a href="#">[MS-LSAT]: Local Security Authority (Translation Methods) Remote Protocol</a>	<p>Specifies the Local Security Authority (Translation Methods) Remote Protocol, which is implemented in Windows-based products to translate identifiers for security principal between human-readable and machine-readable forms.</p> <p><a href="#">Click here to view this version of the [MS-LSAT] PDF.</a></p>
<a href="#">[MS-LWSSP]: Lightweight Web Services Security Profile</a>	<p>Specifies the Lightweight Web Services Security Profile. This profile specifies how to perform lightweight client authentication and security token exchange based on set of security-related Web services protocols.</p> <p><a href="#">Click here to view this version of the [MS-LWSSP] PDF.</a></p>
<a href="#">[MS-MAIL]: Remote Mailslot Protocol</a>	<p>Specifies the Remote Mailslot Protocol. This protocol is a simple, nonsecure, and unidirectional interprocess communications (IPC) protocol between a client and server.</p> <p><a href="#">Click here to view this version of the [MS-MAIL] PDF.</a></p>
<a href="#">[MS-MCIS]: Content Indexing Services Protocol</a>	<p>Specifies the Content Indexing Services Protocol, which enables a client to communicate with a server hosting an indexing service to issue queries.</p> <p><a href="#">Click here to view this version of the [MS-MCIS] PDF.</a></p>
<a href="#">[MS-MDE]: Mobile Device Enrollment Protocol</a>	<p>Specifies the Mobile Device Management Enrollment Protocol, which provides a mechanism for discovering devices and enrolling them into a management system. After enrollment, devices can be managed through the Microsoft Mobile Device Management Protocol [MS-MDM].</p> <p><a href="#">Click here to view this version of the [MS-MDE] PDF.</a></p>

Specification	Description
<a href="#">[MS-MDE2]: Mobile Device Enrollment Protocol Version 2</a>	<p>Specifies version 2 of the Mobile Device Enrollment Protocol (MDE), which enables enrolling a device with the DMS through an Enrollment Service (ES). The protocol includes the discovery of the Management Enrollment Service (MES) and enrollment with the ES.</p> <p><a href="#">Click here to view this version of the [MS-MDE2] PDF.</a></p>
<a href="#">[MS-MDM]: Mobile Device Management Protocol</a>	<p>Specifies the Mobile Device Management Protocol (MDM), a subset of the Open Mobile Association (OMA) standard protocol, which provides a mechanism for managing devices previously enrolled into a management system through the Microsoft Mobile Device Management Enrollment Protocol [MS-MDE].</p> <p><a href="#">Click here to view this version of the [MS-MDM] PDF.</a></p>
<a href="#">[MS-MICE]: Miracast over Infrastructure Connection Establishment Protocol</a>	<p>The Miracast over Infrastructure Connection Establishment Protocol specifies a connection negotiation sequence used to connect, indicate readiness to connect, and disconnect from a Miracast over Infrastructure endpoint. This protocol also specifies the Miracast over Infrastructure Information Element (IE), which helps identify Miracast receivers (sinks) that can support a Miracast session over an infrastructure link (as opposed to a Wi-Fi Direct link).</p> <p><a href="#">Click here to view this version of the [MS-MICE] PDF.</a></p>
<a href="#">[MS-MMSP]: Microsoft Media Server (MMS) Protocol</a>	<p>Specifies the Microsoft Media Server (MMS) Protocol, which defines how MMS streams multimedia from Windows Media Services to Windows Media Player, or to another instance of Windows Media Services. MMS uses TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).</p> <p><a href="#">Click here to view this version of the [MS-MMSP] PDF.</a></p>
<a href="#">[MS-MNPR]: Microsoft NetMeeting Protocol</a>	<p>Specifies the Microsoft NetMeeting Protocol, which implements a method of application sharing over the T.120 Multipoint Communication Service (MCS) layer, using the S20 MCS Channel.</p> <p><a href="#">Click here to view this version of the [MS-MNPR] PDF.</a></p>
<a href="#">[MS-MQBR]: Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm</a>	<p>Specifies the Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm, which is used by MSMQ to communicate across both connected networks and heterogeneous networks.</p> <p><a href="#">Click here to view this version of the [MS-MQBR] PDF.</a></p>

Specification	Description
<a href="#">[MS-MQCN]: Message Queuing (MSMQ): Directory Service Change Notification Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Directory Service Change Notification Protocol. It defines a mechanism used by the MSMQ Directory Service or a queue manager to notify a queue manager of changes to its owned objects.</p> <p><a href="#">Click here to view this version of the [MS-MQCN] PDF.</a></p>
<a href="#">[MS-MQDMPR]: Message Queuing (MSMQ): Common Data Model and Processing Rules</a>	<p>Specifies the Message Queuing (MSMQ): Data Structures, which define an abstract data model and events shared by multiple MSMQ protocols.</p> <p><a href="#">Click here to view this version of the [MS-MQDMPR] PDF.</a></p>
<a href="#">[MS-MQDS]: Message Queuing (MSMQ): Directory Service Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Directory Service Protocol, an RPC-based protocol that is used by MSMQ clients and Message Queuing servers to remotely access and maintain MSMQ directory objects.</p> <p><a href="#">Click here to view this version of the [MS-MQDS] PDF.</a></p>
<a href="#">[MS-MQDSSM]: Message Queuing (MSMQ): Directory Service Schema Mapping</a>	<p>Specifies the Message Queuing (MSMQ): Data Structures that are used by any protocol that manipulates the subset of the abstract data elements and data element attributes defined in [MS-MQDMPR] section 3.1.</p> <p><a href="#">Click here to view this version of the [MS-MQDSSM] PDF.</a></p>
<a href="#">[MS-MQMP]: Message Queuing (MSMQ): Queue Manager Client Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Queue Manager Client Protocol, which enables communication between message queuing client applications and an MSMQ Queue Manager.</p> <p><a href="#">Click here to view this version of the [MS-MQMP] PDF.</a></p>
<a href="#">[MS-MMQ]: Message Queuing (MSMQ): Data Structures</a>	<p>Specifies Message Queuing (MSMQ): Data Structures, which contains common definitions and data structures that are used in the Microsoft Message Queuing protocols.</p> <p><a href="#">Click here to view this version of the [MS-MMQ] PDF.</a></p>
<a href="#">[MS-MQMR]: Message Queuing (MSMQ): Queue Manager Management Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Queue Manager Management Protocol that is used for management operations on the MSMQ server, including monitoring the MSMQ installation and the queues.</p> <p><a href="#">Click here to view this version of the [MS-MQMR] PDF.</a></p>

Specification	Description
<a href="#">[MS-MQQB]: Message Queuing (MSMQ): Message Queuing Binary Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Binary Reliable Messaging Protocol, which defines a mechanism for reliably transferring messages between two message queues located on two different hosts.</p> <p><a href="#">Click here to view this version of the [MS-MQQB] PDF.</a> ↗</p>
<a href="#">[MS-MQQP]: Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol, an RPC-based protocol used by the queue manager and runtime library to read and purge messages from a remote queue.</p> <p><a href="#">Click here to view this version of the [MS-MQQP] PDF.</a> ↗</p>
<a href="#">[MS-MQRR]: Message Queuing (MSMQ): Queue Manager Remote Read Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Queue Manager Remote Read Protocol, an RPC-based protocol that is used by MSMQ clients to read or reject a message from a queue, move a message between queues, and purge messages from a queue.</p> <p><a href="#">Click here to view this version of the [MS-MQRR] PDF.</a> ↗</p>
<a href="#">[MS-MQSD]: Message Queuing (MSMQ): Directory Service Discovery Protocol</a>	<p>Specifies the Message Queuing (MSMQ): Directory Service Discovery Protocol, which is used by MSMQ clients to discover an accessible executing instance of an MSMQ Directory Service server.</p> <p><a href="#">Click here to view this version of the [MS-MQSD] PDF.</a> ↗</p>
<a href="#">[MS-MSB]: Media Stream Broadcast (MSB) Protocol</a>	<p>Specifies the Media Stream Broadcast (MSB) Protocol, which enables distribution of Advanced Systems Format (ASF) packets over a network for which Internet Protocol (IP) multicasting is enabled.</p> <p><a href="#">Click here to view this version of the [MS-MSB] PDF.</a> ↗</p>
<a href="#">[MS-MSBD]: Media Stream Broadcast Distribution (MSBD) Protocol</a>	<p>Specifies the Media Stream Broadcast Distribution (MSBD) Protocol, which describes how to transfer an audio-visual content stream from a server to a single client.</p> <p><a href="#">Click here to view this version of the [MS-MSBD] PDF.</a> ↗</p>
<a href="#">[MS-MSRP]: Messenger Service Remote Protocol</a>	<p>Specifies the Messenger Service Remote Protocol, a set of RPC interfaces that instructs a server to display short text messages to a console user, to deliver messages to a local or remote server for display to a console user, and to manage the names for which the server receives messages.</p> <p><a href="#">Click here to view this version of the [MS-MSRP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-MWBE]: Microsoft Web Browser Federated Sign-On Protocol Extensions</a>	<p>Specifies the Microsoft Web Browser Federated Sign-On Protocol Extensions. This extension enables Web browser requestors that do not support scripting (to create POST messages) and enables passing security identifiers (SIDs) in Security Assertion Markup Language (SAML) V1.1 assertions. It is assumed that the reader is familiar with the terms, concepts, and protocols that are defined in [MS-MWBF].</p> <p><a href="#">Click here to view this version of the [MS-MWBE] PDF.</a> ↗</p>
<a href="#">[MS-MWBF]: Microsoft Web Browser Federated Sign-On Protocol</a>	<p>Specifies the Microsoft Web Browser Federated Sign-On Protocol, which is primarily a restriction of the protocol that is specified in [WSFederation1.2] section 13. The restrictions are designed to enable greater interoperability by reducing the number of variations that must be implemented. This protocol also specifies minor additions to [WSFederation1.2] section 13 to handle common scenarios.</p> <p><a href="#">Click here to view this version of the [MS-MWBF] PDF.</a> ↗</p>
<a href="#">[MS-N2HT]: Negotiate and Nego2 HTTP Authentication Protocol</a>	<p>Specifies the Negotiate and Nego2 HTTP Authentication Protocol, which describes support for SPNEGO authentication as specified in [RFC4559]. The tokens are transmitted using base64-encoding. This protocol calls out the differences in the Microsoft implementation from what is specified in [RFC4559], where applicable.</p> <p><a href="#">Click here to view this version of the [MS-N2HT] PDF.</a> ↗</p>
<a href="#">[MS-NBTE]: NetBIOS over TCP (NBT) Extensions</a>	<p>Specifies the NetBIOS over TCP (NBT) Extensions, as specified in [RFC1001] and [RFC1002]. These extensions modify the syntax of allowable NetBIOS names and the behavior of timers, and add support for multihomed hosts.</p> <p><a href="#">Click here to view this version of the [MS-NBTE] PDF.</a> ↗</p>
<a href="#">[MS-NCNBI]: Network Controller Northbound Interface</a>	<p>Specifies the Network Controller Protocol, which is used by tenants and network administrators to control data center networking. Common tasks that would use these APIs include designing and monitoring a virtual network in a data center.</p> <p><a href="#">Click here to view this version of the [MS-NCNBI] PDF.</a> ↗</p>
<a href="#">[MS-NCT]: Network Cost Transfer Protocol</a>	<p>Enables an 802.11 wireless access point (AP) to inform a wireless client of the network cost and hints about the AP type.</p> <p><a href="#">Click here to view this version of the [MS-NCT] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-NEGOEX]: SPNEGO Extended Negotiation (NEGOEX) Security Mechanism</a>	<p>Specifies the SPNEGO Extended Negotiation (NEGOEX) Security Mechanism that enhances the capabilities of SPNEGO [RFC4178] by providing a security mechanism which can be negotiated by SPNEGO. When the NEGOEX security mechanism is selected by SPNEGO, NEGOEX provides a method allowing selection of a common authentication protocol based on meta-data such as trust configurations.</p> <p><a href="#">Click here to view this version of the [MS-NEGOEX] PDF.</a> ↗</p>
<a href="#">[MS-NETTR]: .NET Tracing Protocol</a>	<p>Specifies the .NET Tracing Protocol, which provides a method for correlating call traces in a .NET remoting application.</p> <p><a href="#">Click here to view this version of the [MS-NETTR] PDF.</a> ↗</p>
<a href="#">[MS-NFPB]: Near Field Proximity: Bidirectional Services Protocol</a>	<p>Specifies the Near Field Proximity: Bidirectional Services Protocol, which provides a way for devices to discover services and versions from one device to another. The protocol uses the ""Proximity Publication Subscription"" transport to exchange messages between peers.</p> <p><a href="#">Click here to view this version of the [MS-NFPB] PDF.</a> ↗</p>
<a href="#">[MS-NFPS]: Near Field Proximity: Sharing Protocol</a>	<p>Specifies the Near Field Proximity: Sharing Protocol, which provides a way for devices to share files over an already established single-purpose channel. A client can use this protocol to send a set of files packaged in an Open Packaging Convention (OPC) file and encrypted over the channel.</p> <p><a href="#">Click here to view this version of the [MS-NFPS] PDF.</a> ↗</p>
<a href="#">[MS-NKPU]: Network Key Protector Unlock Protocol</a>	<p>Specifies the Network Key Protector Unlock Protocol, which enables a client to send an encrypted package of key material along with a session key to a remote server and to receive the decrypted key material protected by the session key.</p> <p><a href="#">Click here to view this version of the [MS-NKPU] PDF.</a> ↗</p>
<a href="#">[MS-NLMP]: NT LAN Manager (NTLM) Authentication Protocol</a>	<p>Specifies the NT LAN Manager (NTLM) Authentication Protocol, used in Windows for authentication between clients and servers. NTLM is used by application protocols to authenticate remote users and, optionally, to provide session security when requested by the application.</p> <p><a href="#">Click here to view this version of the [MS-NLMP] PDF.</a> ↗</p>
<a href="#">[MS-NMFMB]: .NET Message Framing MSMQ Binding Protocol</a>	<p>Specifies the .NET Message Framing MSMQ Binding Protocol, which defines how the mechanism described in [MC-NMF] for framing messages over any transport protocol can be applied over Message Queue (MSMQ). This protocol specification also defines how to indicate the use of .NET Message Framing over MSMQ as a SOAP transport in Web Services Description Language (WSDL).</p> <p><a href="#">Click here to view this version of the [MS-NMFMB] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-NMFTB]: .NET Message Framing TCP Binding Protocol</a>	<p>Specifies how the .NET Message Framing Protocol [MC-NMF] is bound to a TCP connection, including the initiation of the stream by using the net.tcp URI scheme and the application of .NET Message Framing over TCP as a SOAP transport in WSDL.</p> <p><a href="#">Click here to view this version of the [MS-NMFTB] PDF.</a></p>
<a href="#">[MS-NNS]: .NET NegotiateStream Protocol</a>	<p>Specifies the .NET NegotiateStream Protocol, which provides mutually authenticated and confidential communication over a TCP connection. It uses the Simple and Protected GSS-API Negotiation mechanism (SPNEGO) for security services (authentication, key derivation, and data encryption and decryption).</p> <p><a href="#">Click here to view this version of the [MS-NNS] PDF.</a></p>
<a href="#">[MS-NNTP]: NT LAN Manager (NTLM) Authentication: Network News Transfer Protocol (NNTP) Extension</a>	<p>Specifies the NT LAN Manager (NTLM) Authentication: Network News Transfer Protocol (NNTP) Extension, which defines the use of NTLM authentication by NNTP to facilitate client authentication to a Windows-based NNTP server.</p> <p><a href="#">Click here to view this version of the [MS-NNTP] PDF.</a></p>
<a href="#">[MS-NRBF]: .NET Remoting: Binary Format Data Structure</a>	<p>Specifies the .NET Remoting: Binary Format Data Structure protocol, which defines a set of structures for representing object graph or method invocation information as an octet stream.</p> <p><a href="#">Click here to view this version of the [MS-NRBF] PDF.</a></p>
<a href="#">[MS-NRLS]: .NET Remoting: Lifetime Services Extension</a>	<p>Specifies the .NET Remoting: Lifetime Services Extension, which adds lifetime and remote activation capabilities to the .NET Remoting Core Protocol (specified in [MS-NRTP]).</p> <p><a href="#">Click here to view this version of the [MS-NRLS] PDF.</a></p>
<a href="#">[MS-NRPC]: Netlogon Remote Protocol</a>	<p>Specifies the Netlogon Remote Protocol, an RPC interface that is used for user and machine authentication on domain-based networks; to replicate the user account database for operating systems earlier than Windows 2000 backup domain controllers; to discover, manage, and maintain domain relationships of domain members and domain controllers across domains.</p> <p><a href="#">Click here to view this version of the [MS-NRPC] PDF.</a></p>
<a href="#">[MS-NRTP]: .NET Remoting: Core Protocol</a>	<p>Specifies the .NET Remoting: Core Protocol, a mechanism by which a calling program can invoke a method in a different address space over the network. Arguments are passed along as part of the invocation message, and return values are sent in the response.</p> <p><a href="#">Click here to view this version of the [MS-NRTP] PDF.</a></p>

Specification	Description
<a href="#">[MS-NSPI]: Name Service Provider Interface (NSPI) Protocol</a>	<p>Specifies the Name Service Provider Interface (NSPI) Protocol, which provides messaging clients with a way to access and manipulate addressing data stored by a server. This protocol consists of an abstract data model and a single RPC call interface to manipulate data in that model.</p> <p><a href="#">Click here to view this version of the [MS-NSPI] PDF.</a></p>
<a href="#">[MS-NTHT]: NTLM Over HTTP Protocol</a>	<p>Specifies the NTLM Over HTTP Protocol, which is used to authenticate a Web client to a Web server. This protocol authentication variant works only with NTLM; the Kerberos protocol is not supported.</p> <p><a href="#">Click here to view this version of the [MS-NTHT] PDF.</a></p>
<a href="#">[MS-NVGREE]: Network Virtualization using Generic Routing Encapsulation (NVGRE) Extensions</a>	<p>Specifies the Network Virtualization using Generic Routing Encapsulation (NVGRE) Extensions protocol, which adds additional support to the NVGRE protocol by defining two new extension/control messages. One message initiates traffic redirection to a new network virtualization endpoint (NVE) when a VM is moved to a new NVE during a data center migration event. The other new message pushes out a policy refresh via an NVE in response to the moved VM.</p> <p><a href="#">Click here to view this version of the [MS-NVGREE] PDF.</a></p>
<a href="#">[MS-OAPX]: OAuth 2.0 Protocol Extensions</a>	<p>Specifies the OAuth 2.0 Protocol Extensions, which are used to extend the OAuth 2.0 Authorization Framework. These extensions enable authorization features such as resource specification, request identifiers, and login hints.</p> <p><a href="#">Click here to view this version of the [MS-OAPX] PDF.</a></p>
<a href="#">[MS-OAPXBC]: OAuth 2.0 Protocol Extensions for Broker Clients</a>	<p>Specifies the OAuth 2.0 Protocol Extensions for Broker Clients, extensions to [RFC6749] (The OAuth 2.0 Authorization Framework) that allow a broker client to obtain access tokens on behalf of calling clients.</p> <p><a href="#">Click here to view this version of the [MS-OAPXBC] PDF.</a></p>
<a href="#">[MS-OAUT]: OLE Automation Protocol</a>	<p>Specifies the OLE Automation Protocol, which uses DCOM as its transport layer and provides support for an additional set of types as well as for a late-bound calling mechanism.</p> <p><a href="#">Click here to view this version of the [MS-OAUT] PDF.</a></p>
<a href="#">[MS-OCSP]: Online Certificate Status Protocol (OCSP) Extensions</a>	<p>Specifies the Online Certificate Status Protocol (OCSP) Extensions, which defines the data that needs to be exchanged between an application that checks the status of a certificate and the responder that provides the status.</p> <p><a href="#">Click here to view this version of the [MS-OCSP] PDF.</a></p>

Specification	Description
<a href="#">[MS-OCSPA]: Microsoft OCSP Administration Protocol</a>	<p>Specifies the Microsoft OCSP Administration Protocol, which consists of a set of distributed component object model (DCOM) interfaces that allows administrative tools to configure the properties of the Online Responder.</p> <p><a href="#">Click here to view this version of the [MS-OCSPA] PDF.</a></p>
<a href="#">[MS-ODATA]: Open Data Protocol (OData)</a>	<p>Specifies the Open Data (OData) Protocol. This protocol enables applications to expose data, by using common Web technologies, and by means of a data service that can be consumed by clients within corporate networks and across the Internet.</p> <p><a href="#">Click here to view this version of the [MS-ODATA] PDF.</a></p>
<a href="#">[MS-OIDCE]: OpenID Connect 1.0 Protocol Extensions</a>	<p>Specifies the OpenID Connect 1.0 Protocol Extensions. These extensions define additional claims to carry information about the end user, including the user principal name, a locally unique identifier, a time for password expiration, and a URL for password change. These extensions also define additional provider metadata that enable the discovery of the issuer of access tokens and give additional information about provider capabilities.</p> <p><a href="#">Click here to view this version of the [MS-OIDCE] PDF.</a></p>
<a href="#">[MS-OLEDS]: Object Linking and Embedding (OLE) Data Structures</a>	<p>Specifies the Object Linking and Embedding (OLE) Data Structures. These structures enable applications to create documents that contain linked or embedded objects.</p> <p><a href="#">Click here to view this version of the [MS-OLEDS] PDF.</a></p>
<a href="#">[MS-OLEPS]: Object Linking and Embedding (OLE) Property Set Data Structures</a>	<p>Specifies the Object Linking and Embedding (OLE): Property Set Data Structures. These structures enable applications to write metadata in a manner that is discoverable to other software.</p> <p><a href="#">Click here to view this version of the [MS-OLEPS] PDF.</a></p>
<a href="#">[MS-OTPCE]: One-Time Password Certificate Enrollment Protocol</a>	<p>Specifies the One-Time Password Certificate Enrollment Protocol, which enhances network security in remote access connections by utilizing different components, such as the one-time password (OTP) authentication mechanism as well as a short-lived smart card logon certificate.</p> <p><a href="#">Click here to view this version of the [MS-OTPCE] PDF.</a></p>
<a href="#">[MS-PAC]: Privilege Attribute Certificate Data Structure</a>	<p>Specifies the Privilege Attribute Certificate Data Structure, which is used to encode authorization information. The Privilege Attribute Certificate also contains memberships, additional credential information, profile and policy information, and supporting security metadata.</p> <p><a href="#">Click here to view this version of the [MS-PAC] PDF.</a></p>

Specification	Description
<a href="#">[MS-PAN]: Print System Asynchronous Notification Protocol</a>	<p>Specifies the [MS-PAN]: Print System Asynchronous Notification Protocol, an asynchronous protocol that clients use to receive print status notifications from a print server and send server-requested responses to those notifications back to the server. It is based on the Remote Procedure Call (RPC) protocol, as specified in [C706].</p> <p><a href="#">Click here to view this version of the [MS-PAN] PDF.</a> ↗</p>
<a href="#">[MS-PAR]: Print System Asynchronous Remote Protocol</a>	<p>Specifies the Print System Asynchronous Remote Protocol, which defines the communication of print job processing and print system management information between a print client and a print server.</p> <p><a href="#">Click here to view this version of the [MS-PAR] PDF.</a> ↗</p>
<a href="#">[MS-PASS]: Passport Server Side Include (SSI) Version 1.4 Protocol</a>	<p>Specifies the Passport Server Side Include (SSI) Version 1.4 Protocol, which describes how messages are encapsulated on the wire.</p> <p><a href="#">Click here to view this version of the [MS-PASS] PDF.</a> ↗</p>
<a href="#">[MS-PBSD]: Publication Services Data Structure</a>	<p>Specifies the Publication Services Data Structure. This structure describes the data that computers use to describe themselves and the resources they offer as Web services over IP-based networks.</p> <p><a href="#">Click here to view this version of the [MS-PBSD] PDF.</a> ↗</p>
<a href="#">[MS-PCCRC]: Peer Content Caching and Retrieval: Content Identification</a>	<p>Specifies Peer Content Caching and Retrieval: Content Identification, the content information format used by the Windows Branch Caching Framework to uniquely identify content for discovery and retrieval purposes.</p> <p><a href="#">Click here to view this version of the [MS-PCCRC] PDF.</a> ↗</p>
<a href="#">[MS-PCCRD]: Peer Content Caching and Retrieval: Discovery Protocol</a>	<p>Specifies the Peer Content Caching and Retrieval Discovery Protocol, which is based on the Web Services Dynamic Discovery (WS-Discovery) protocol. It is a content caching and retrieval framework based on a peer-to-peer discovery and distribution model.</p> <p><a href="#">Click here to view this version of the [MS-PCCRD] PDF.</a> ↗</p>
<a href="#">[MS-PCCRR]: Peer Content Caching and Retrieval: Retrieval Protocol</a>	<p>Specifies the Peer Content Caching and Retrieval: Retrieval Protocol. This protocol defines two message exchanges, one for querying the server for the availability of certain content, and the other for retrieving content from a server.</p> <p><a href="#">Click here to view this version of the [MS-PCCRR] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-PCC RTP]: Peer Content Caching and Retrieval: Hypertext Transfer Protocol (HTTP) Extensions</a>	<p>Specifies the Peer Content Caching and Retrieval: Hypertext Transfer Protocol (HTTP) Extensions, which implements a new type of content encoding, PeerDist, that can be used in HTTP/1.1. In particular, it specifies the mechanism used by an HTTP/1.1 client and an HTTP/1.1 server to communicate with each other using the PeerDist content encoding.</p> <p><a href="#">Click here to view this version of the [MS-PCC RTP] PDF.</a> ↗</p>
<a href="#">[MS-PCHC]: Peer Content Caching and Retrieval: Hosted Cache Protocol</a>	<p>Specifies the Peer Content Caching and Retrieval: Hosted Cache Protocol, used by clients to offer metadata to a hosted cache server.</p> <p><a href="#">Click here to view this version of the [MS-PCHC] PDF.</a> ↗</p>
<a href="#">[MS-PCQ]: Performance Counter Query Protocol</a>	<p>Specifies the Performance Counter Query Protocol, which is used for browsing performance counters and retrieving performance counter values from a server.</p> <p><a href="#">Click here to view this version of the [MS-PCQ] PDF.</a> ↗</p>
<a href="#">[MS-PEAP]: Protected Extensible Authentication Protocol (PEAP)</a>	<p>Specifies the Protected Extensible Authentication Protocol (PEAP), which adds security services to the Extensible Authentication Protocol methods.</p> <p><a href="#">Click here to view this version of the [MS-PEAP] PDF.</a> ↗</p>
<a href="#">[MS-PKAP]: Public Key Authentication Protocol</a>	<p>Specifies the Public Key Authentication Protocol, which provides a method for HTTP clients to prove possession of a private key to a web server without having to rely on client Transport Layer Security (TLS) support from the underlying platform.</p> <p><a href="#">Click here to view this version of the [MS-PKAP] PDF.</a> ↗</p>
<a href="#">[MS-PKCA]: Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol</a>	<p>Specifies the Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol. This protocol enables the use of public key cryptography in the initial authentication exchange of the Kerberos Protocol (PKINIT) and specifies the Windows implementation of PKINIT where it differs from [RFC4556].</p> <p><a href="#">Click here to view this version of the [MS-PKCA] PDF.</a> ↗</p>
<a href="#">[MS-PLA]: Performance Logs and Alerts Protocol</a>	<p>Specifies the Performance Logs and Alerts Protocol, which provides a set of DCOM interfaces to control data collection on a remote system. The control includes starting, stopping, scheduling, and configuration of data collector objects, and the creation of alerts.</p> <p><a href="#">Click here to view this version of the [MS-PLA] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-PNRP]: Peer Name Resolution Protocol (PNRP) Version 4.0</a>	<p>Specifies the Peer Name Resolution Protocol (PNRP) Version 4.0, which is used to resolve a name to a set of information, such as IP addresses; to maintain a cloud of peer nodes; to maintain a distributed cache of endpoint information; and to transfer requests for Peer Name resolutions between nodes.</p> <p><a href="#">Click here to view this version of the [MS-PNRP] PDF.</a></p>
<a href="#">[MS-POP3]: NT LAN Manager (NTLM) Authentication: Post Office Protocol - Version 3 (POP3) Extension</a>	<p>Specifies the NT LAN Manager (NTLM) Authentication: Post Office Protocol - Version 3 (POP3) Extension, which describes the use of NTLM Authentication (see [MS-NLMP]) by the Post Office Protocol 3 (POP3) to facilitate client authentication to a Windows POP3 server. POP3 specifies a protocol for the inquiry and retrieval of electronic mail.</p> <p><a href="#">Click here to view this version of the [MS-POP3] PDF.</a></p>
<a href="#">[MS-PPGRH]: Peer-to-Peer Graphing Protocol</a>	<p>Specifies the Peer-to-Peer Graphing Protocol, a peer-to-peer protocol for establishing and maintaining a connected set of nodes (referred to as a graph), and replicating data among the nodes.</p> <p><a href="#">Click here to view this version of the [MS-PPGRH] PDF.</a></p>
<a href="#">[MS-PPPI]: PPP Over IrDA Dialup Protocol</a>	<p>Specifies the PPP Over IrDA Dialup Protocol, which enables the scenario in which a computer with infrared capabilities obtains network access by using a modem via the infrared link.</p> <p><a href="#">Click here to view this version of the [MS-PPPI] PDF.</a></p>
<a href="#">[MS-PPSEC]: Peer-to-Peer Grouping Security Protocol</a>	<p>Specifies the Peer-to-Peer Grouping Security Protocol (P2P Grouping), which layers on top of the Peer-to-Peer Graphing Protocol [MS-PPGRH] and adds security and discovery services.</p> <p><a href="#">Click here to view this version of the [MS-PPSEC] PDF.</a></p>
<a href="#">[MS-PROPSTORE]: Property Store Binary File Format</a>	<p>Specifies the Property Store Binary File Format. This file format is a persistence format for a set of properties. Implementers can use this file format to store a set of properties in a file or within another structure.</p> <p><a href="#">Click here to view this version of the [MS-PROPSTORE] PDF.</a></p>
<a href="#">[MS-PSDP]: Proximity Service Discovery Protocol</a>	<p>Specifies the Proximity Service Discovery Protocol, which conveys service discovery information, such as service advertisements, as part of Beacon frames, as specified in [IEEE802.11-2007].</p> <p><a href="#">Click here to view this version of the [MS-PSDP] PDF.</a></p>

Specification	Description
<a href="#">[MS-PSRDP]: PowerShell Remote Debugging Protocol</a>	<p>Specifies the PowerShell Remote Debugging Protocol (PSRDP). This protocol extends the existing PowerShell Remoting Protocol (PSRP) specified in [MS-PSRP] to support debugging over a remote session.</p> <p><a href="#">Click here to view this version of the [MS-PSRDP] PDF.</a> ↗</p>
<a href="#">[MS-PSRP]: PowerShell Remoting Protocol</a>	<p>Specifies the Windows PowerShell Remoting Protocol, which encodes messages prior to sending them over the Web Services Management Protocol Extensions for the Windows Vista [MS-WSMV] layer.</p> <p><a href="#">Click here to view this version of the [MS-PSRP] PDF.</a> ↗</p>
<a href="#">[MS-PTPT]: Point-to-Point Tunneling Protocol (PPTP) Profile</a>	<p>Specifies the Point-to-Point Tunneling Protocol, which allows the Point-to-Point Protocol (PPP) [RFC1661] to be tunneled through an IP network.</p> <p><a href="#">Click here to view this version of the [MS-PTPT] PDF.</a> ↗</p>
<a href="#">[MS-QDP]: Quality Windows Audio/Video Experience (qWave): Wireless Diagnostics Protocol</a>	<p>Specifies the Quality Windows Audio/Video Experience (qWave): Wireless Diagnostics Protocol. This protocol is used to obtain information from a host or a device about its wireless characteristics, which can facilitate the diagnosis of wireless network issues.</p> <p><a href="#">Click here to view this version of the [MS-QDP] PDF.</a> ↗</p>
<a href="#">[MS-QLPB]: Quality Windows Audio/Video Experience (qWave): Layer 3 Probing Protocol</a>	<p>Specifies the Quality Windows Audio/Video Experience (qWave): Layer 3 Probing (L3P) (qWave) Protocol, which operates over TCP/IP and UDP/IP. qWave enables applications to evaluate link bandwidth and quality by analyzing timestamps of probe packets transmitted between two devices.</p> <p><a href="#">Click here to view this version of the [MS-QLPB] PDF.</a> ↗</p>
<a href="#">[MS-RA]: Remote Assistance Protocol</a>	<p>Specifies the Remote Assistance Protocol, which is used after a remote assistance connection is established between two computers.</p> <p><a href="#">Click here to view this version of the [MS-RA] PDF.</a> ↗</p>
<a href="#">[MS-RAA]: Remote Authorization API Protocol</a>	<p>Specifies the Remote Authorization API Protocol, which is used to perform ""what-if"" authorization queries on remote computers. It allows applications to simulate an access control decision that would be made when a principal attempts to access a remote resource protected with an authorization policy.</p> <p><a href="#">Click here to view this version of the [MS-RAA] PDF.</a> ↗</p>

Specification	Description
[MS-RAI]: Remote Assistance Initiation Protocol	<p>Specifies the Remote Assistance Initiation Protocol, which enables an authorized expert to start Remote Assistance (RA) on a remote novice computer to retrieve data that is required to make a Remote Assistance connection from the expert's computer to the novice's computer.</p> <p><a href="#">Click here to view this version of the [MS-RAI] PDF.</a> ↗</p>
[MS-RAIOP]: Remote Assistance Initiation over PNRP Protocol	<p>Specifies the Remote Assistance Initiation over PNRP Protocol, which is used to establish a Remote Assistance connection between two computers.</p> <p><a href="#">Click here to view this version of the [MS-RAIOP] PDF.</a> ↗</p>
[MS-RAIW]: Remote Administrative Interface: WINS	<p>Specifies the Remote Administrative Interface: WINS protocol, which enables local or remote administration of the Windows Internet Name Service (WINS) within the Microsoft Management Console (MMC) WINS snap-in and the NetSh command line (WINS context).</p> <p><a href="#">Click here to view this version of the [MS-RAIW] PDF.</a> ↗</p>
[MS-RAP]: Remote Administration Protocol	<p>Specifies the Microsoft Remote Administration Protocol (RAP), which Microsoft LAN Manager uses to perform remote administrative functions and is included in the Microsoft Windows operating system for compatibility reasons.</p> <p><a href="#">Click here to view this version of the [MS-RAP] PDF.</a> ↗</p>
[MS-RASA]: Remote Access Server Advertisement (RASADV) Protocol	<p>Specifies the Remote Access Server Advertisement (RASADV) Protocol, by which Remote Access Service (RAS) Servers advertise their presence within a local network, enabling network administrators to detect nonmalicious configuration and deployment of gateways providing external access to their network.</p> <p><a href="#">Click here to view this version of the [MS-RASA] PDF.</a> ↗</p>
[MS-RCMP]: Remote Certificate Mapping Protocol	<p>Specifies the Remote Certificate Mapping Protocol, which enables servers to use a directory, database, or other technology to map the user's X.509 certificate to a security principal.</p> <p><a href="#">Click here to view this version of the [MS-RCMP] PDF.</a> ↗</p>
[MS-RDC]: Remote Differential Compression Algorithm	<p>Specifies the Remote Differential Compression Algorithm protocol, which enables efficient synchronization of files with a remote source by using compression techniques to minimize the amount of data sent between a client and server.</p> <p><a href="#">Click here to view this version of the [MS-RDC] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-RDPADRV]: Remote Desktop Protocol: Audio Level and Drive Letter Persistence Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Audio Level and Drive Letter Persistence Virtual Channel Extension, which allows an RDP (remote desktop connection) client device to mimic a Windows client PC session with respect to audio levels and drive letters.</p> <p><a href="#">Click here to view this version of the [MS-RDPADRV] PDF.</a> ↗</p>
<a href="#">[MS-RDPBCGR]: Remote Desktop Protocol: Basic Connectivity and Graphics Remoting</a>	<p>Specifies the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting, designed to facilitate user interaction with a remote computer system by transferring graphics display information from the remote computer to the user and transporting input from the user to the remote computer, where it may be injected locally.</p> <p><a href="#">Click here to view this version of the [MS-RDPBCGR] PDF.</a> ↗</p>
<a href="#">[MS-RDPCR2]: Remote Desktop Protocol: Composited Remoting V2</a>	<p>Specifies the Remote Desktop Protocol: Composited Remoting V2, which displays the contents of the Windows-based desktop running on one machine on a second machine connected to the first via a network.</p> <p><a href="#">Click here to view this version of the [MS-RDPCR2] PDF.</a> ↗</p>
<a href="#">[MS-RDPEA]: Remote Desktop Protocol: Audio Output Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Audio Output Virtual Channel Extension, which transfers audio data from the server to the client.</p> <p><a href="#">Click here to view this version of the [MS-RDPEA] PDF.</a> ↗</p>
<a href="#">[MS-RDPEAI]: Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension, which transfers audio data from a client to a server.</p> <p><a href="#">Click here to view this version of the [MS-RDPEAI] PDF.</a> ↗</p>
<a href="#">[MS-RDPEAR]: Remote Desktop Protocol Authentication Redirection Virtual Channel</a>	<p>Performs authentication over a Remote Desktop connection. By establishing a virtual channel between the source and the target devices, it can relay authentication requests received by the target device to the source device.</p> <p><a href="#">Click here to view this version of the [MS-RDPEAR] PDF.</a> ↗</p>
<a href="#">[MS-RDPECAM]: Remote Desktop Protocol: Video Capture Virtual Channel Extension</a>	<p>The Remote Desktop Protocol: Video Capture Virtual Channel Extension adds remoting of video capture devices, such as webcams, to the Basic Connectivity and Graphics Remoting Protocol.</p> <p><a href="#">Click here to view this version of the [MS-RDPECAM] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-RDPECLIP]: Remote Desktop Protocol: Clipboard Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Clipboard Virtual Channel Extension, which enables users to seamlessly transfer data via the system clipboard between applications that are running on different computers.</p> <p><a href="#">Click here to view this version of the [MS-RDPECLIP] PDF.</a> ↗</p>
<a href="#">[MS-RDPEDC]: Remote Desktop Protocol: Desktop Composition Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Desktop Composition Virtual Channel Extension, which enables a remote display client to replicate the functionality of the Desktop Window Manager (DWM) across a network boundary.</p> <p><a href="#">Click here to view this version of the [MS-RDPEDC] PDF.</a> ↗</p>
<a href="#">[MS-RDPEDISP]: Remote Desktop Protocol: Display Update Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Display Control Virtual Channel Extension to the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting, as specified in [MS-RDPBCGR]. This control protocol is used to request display configuration changes in a remote session.</p> <p><a href="#">Click here to view this version of the [MS-RDPEDISP] PDF.</a> ↗</p>
<a href="#">[MS-RDPEDYC]: Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension, which supports features such as classes of priority (that may be used to implement bandwidth allocation) and individually connected endpoints using dynamic virtual channel (DVC) listeners.</p> <p><a href="#">Click here to view this version of the [MS-RDPEDYC] PDF.</a> ↗</p>
<a href="#">[MS-RDPEECO]: Remote Desktop Protocol: Virtual Channel Echo Extension</a>	<p>Specifies the Remote Desktop Protocol: Virtual Channel Echo Extension. This extension is used as a ping and echo mechanism to determine various network characteristics that are significant for RDP.</p> <p><a href="#">Click here to view this version of the [MS-RDPEECO] PDF.</a> ↗</p>
<a href="#">[MS-RDPEFS]: Remote Desktop Protocol: File System Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: File System Virtual Channel Extension, which runs over a static virtual channel with the name RDPDR.</p> <p><a href="#">Click here to view this version of the [MS-RDPEFS] PDF.</a> ↗</p>
<a href="#">[MS-RDPEGDI]: Remote Desktop Protocol: Graphics Device Interface (GDI) Acceleration Extensions</a>	<p>Specifies the Remote Desktop Protocol: Graphics Device Interface (GDI) Acceleration Extensions, which reduces the bandwidth associated with graphics remoting by encoding the drawing operations that produce an image instead of encoding the actual image.</p> <p><a href="#">Click here to view this version of the [MS-RDPEGDI] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-RDPEGFX]: Remote Desktop Protocol: Graphics Pipeline Extension</a>	<p>Specifies the Remote Desktop Protocol: Graphics Pipeline Extension, a graphics protocol that is used to encode graphics display data generated in a remote terminal server session so that the data can be sent from the server and received, decoded, and rendered by a compatible client. The net effect is that a desktop or an application running on a remote terminal server appears as if it is running locally.</p> <p><a href="#">Click here to view this version of the [MS-RDPEGFX] PDF.</a></p>
<a href="#">[MS-RDPEGT]: Remote Desktop Protocol: Geometry Tracking Virtual Channel Protocol Extension</a>	<p>Specifies the Remote Desktop Protocol: Geometry Tracking Virtual Channel Extension, which extends the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting. This protocol facilitates graphics rendering between a desktop host and a remote desktop client in a way that the client does not need to know the origin of the graphics.</p> <p><a href="#">Click here to view this version of the [MS-RDPEGT] PDF.</a></p>
<a href="#">[MS-RDPEI]: Remote Desktop Protocol: Input Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Input Virtual Channel Extension, which is used to remote multitouch input frames from a terminal server client to a terminal server. Multitouch input frames are generated at the client, encoded, and sent to the server. Thereafter, these frames are received and decoded by the server and injected into the session associated with the remote user.</p> <p><a href="#">Click here to view this version of the [MS-RDPEI] PDF.</a></p>
<a href="#">[MS-RDPELE]: Remote Desktop Protocol: Licensing Extension</a>	<p>Specifies the Remote Desktop Protocol: Licensing Extension, which expands on the licensing protocol sequence specified in [MS-RDPBCGR] to address scenarios requiring the exchange of licensing information between the client and server.</p> <p><a href="#">Click here to view this version of the [MS-RDPELE] PDF.</a></p>
<a href="#">[MS-RDPEMC]: Remote Desktop Protocol: Multiparty Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Multiparty Virtual Channel Extension, which describes the messages that are exchanged between a remote desktop host and the participants with whom it is engaging in multiparty application sharing.</p> <p><a href="#">Click here to view this version of the [MS-RDPEMC] PDF.</a></p>
<a href="#">[MS-RDPEMT]: Remote Desktop Protocol: Multitransport Extension</a>	<p>This document specifies the Remote Desktop Protocol: Multitransport Extension, which is used to create multiple data-transport connections between an RDP client and an RDP server.</p> <p><a href="#">Click here to view this version of the [MS-RDPEMT] PDF.</a></p>

Specification	Description
[MS-RDPEPC]: Remote Desktop Protocol: Print Virtual Channel Extension	<p>Specifies the Desktop Protocol: Print Virtual Channel Extensions, which specifies the communication used to enable the redirection of printers between a terminal client and a terminal server.</p> <p><a href="#">Click here to view this version of the [MS-RDPEPC] PDF.</a> ↗</p>
[MS-RDPEPNP]: Remote Desktop Protocol: Plug and Play Devices Virtual Channel Extension	<p>Specifies the Remote Desktop Protocol: Plug and Play Devices Virtual Channel Extension, which is used to redirect Plug and Play devices from a terminal client to the terminal server.</p> <p><a href="#">Click here to view this version of the [MS-RDPEPNP] PDF.</a> ↗</p>
[MS-RDPEPS]: Remote Desktop Protocol: Session Selection Extension	<p>Specifies the Remote Desktop Protocol: Session Selection Extension, which expands upon the original connectivity options specified in [MS-RDPBCGR] to address a wide range of new scenarios.</p> <p><a href="#">Click here to view this version of the [MS-RDPEPS] PDF.</a> ↗</p>
[MS-RDPERP]: Remote Desktop Protocol: Remote Programs Virtual Channel Extension	<p>Specifies the Remote Desktop Protocol: Remote Programs Virtual Channel Extension, an RDP feature that presents a remote application (running remotely on a RAIL server) as a local user application (running on the RAIL client machine).</p> <p><a href="#">Click here to view this version of the [MS-RDPERP] PDF.</a> ↗</p>
[MS-RDPESC]: Remote Desktop Protocol: Smart Card Virtual Channel Extension	<p>Specifies the Remote Desktop Protocol: Smart Card Virtual Channel Extension, an extension (including virtual channels) that supports smart card reader-like devices.</p> <p><a href="#">Click here to view this version of the [MS-RDPESC] PDF.</a> ↗</p>
[MS-RDPESP]: Remote Desktop Protocol: Serial and Parallel Port Virtual Channel Extension	<p>Specifies the Remote Desktop Protocol: Serial and Parallel Port Virtual Channel Extension, which redirects serial and parallel ports from a terminal client to the terminal server. This extension allows the server to access client ports as if the connected devices were local to the server.</p> <p><a href="#">Click here to view this version of the [MS-RDPESP] PDF.</a> ↗</p>
[MS-RDPET]: Remote Desktop Protocol: Telemetry Virtual Channel Extension	<p>Specifies the Remote Desktop Protocol: Telemetry Virtual Channel Extension, which extends the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting [MS-RDPBCGR]. This extension is a telemetry protocol that is used to send client performance metrics to the server.</p> <p><a href="#">Click here to view this version of the [MS-RDPET] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-RDPEUDP]: Remote Desktop Protocol: UDP Transport Extension</a>	<p>Specifies the Remote Desktop Protocol: UDP Transport Extension, which extends the transport mechanisms in the Remote Desktop Protocol (RDP) to enable network connectivity between the user's machine and a remote computer system over the User Datagram Protocol (UDP).</p> <p><a href="#">Click here to view this version of the [MS-RDPEUDP] PDF.</a></p>
<a href="#">[MS-RDPEUDP2]: Remote Desktop Protocol: UDP Transport Extension Version 2</a>	<p>Remote Desktop Protocol: UDP Transport Extension Version 2 is used to exchange data, for example audio and video, between a remote desktop client and remote desktop server over UDP transport using a URCP based rate control.</p> <p><a href="#">Click here to view this version of the [MS-RDPEUDP2] PDF.</a></p>
<a href="#">[MS-RDPEUSB]: Remote Desktop Protocol: USB Devices Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: USB Devices Virtual Channel Extension, which is used to redirect USB devices from a terminal client to the terminal server. This allows the server access to devices that are physically connected to the client as if the device were local to the server.</p> <p><a href="#">Click here to view this version of the [MS-RDPEUSB] PDF.</a></p>
<a href="#">[MS-RDPEV]: Remote Desktop Protocol: Video Redirection Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Video Redirection Virtual Channel Extension, which redirects audio/video streams from the terminal server to the terminal client.</p> <p><a href="#">Click here to view this version of the [MS-RDPEV] PDF.</a></p>
<a href="#">[MS-RDPEVOR]: Remote Desktop Protocol: Video Optimized Remoting Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: Video Optimized Remoting Virtual Channel Extension. This is an extension of the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting protocol [MS-RDPBCGR], which runs over a dynamic virtual channel, as specified in [MS-RPVEDYC]. The Remote Desktop Protocol: Video Optimized Remoting Virtual Channel Extension is used to redirect certain rapidly changing graphics content as a video stream from the remote desktop host to the remote desktop client. This protocol specifies the communication between a remote desktop host and a remote desktop client.</p> <p><a href="#">Click here to view this version of the [MS-RDPEVOR] PDF.</a></p>
<a href="#">[MS-RDPEWA]: Remote Desktop Protocol: WebAuthn Virtual Channel Protocol</a>	<p>Specifies the Remote Desktop Protocol (RDP): WebAuthn Virtual Channel Protocol which provides a way for a user to do WebAuthn operations over the RDP protocol. It enables a server to send webauthn request to a client, the client can then use this request to talk to authenticators (platform as well as cross-platform) and reply with the response.</p> <p><a href="#">Click here to view this version of the [MS-RDPEWA] PDF.</a></p>

Specification	Description
<a href="#">[MS-RDPEXPS]: Remote Desktop Protocol: XML Paper Specification (XPS) Print Virtual Channel Extension</a>	<p>Specifies the Remote Desktop Protocol: XML Paper Specification (XPS) Print Virtual Channel Extension, which redirects printing jobs from the terminal server to the terminal client.</p> <p><a href="#">Click here to view this version of the [MS-RDPEXPS] PDF.</a> ↗</p>
<a href="#">[MS-RDPNSC]: Remote Desktop Protocol: NSCodec Extension</a>	<p>Specifies the Remote Desktop Protocol: NSCodec Extension, an extension to the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting (as specified in [MS-RDPBCGR]). This extension specifies an image codec that can be used to encode screen images by utilizing efficient and effective compression.</p> <p><a href="#">Click here to view this version of the [MS-RDPNSC] PDF.</a> ↗</p>
<a href="#">[MS-RDPRFX]: Remote Desktop Protocol: RemoteFX Codec Extension</a>	<p>Specifies the Remote Desktop Protocol: RemoteFX Codec Extension, which uses a lossy image codec to encode screen images with efficient and effective compression.</p> <p><a href="#">Click here to view this version of the [MS-RDPRFX] PDF.</a> ↗</p>
<a href="#">[MS-RDWR]: Remote Desktop Workspace Runtime Protocol</a>	<p>Specifies the Remote Desktop Workspace Runtime Protocol, an HTTP-based protocol for the Remote Desktop Service to discover disconnected sessions for a user and obtain the files required to reconnect to those disconnected sessions. The protocol uses a SOAP-based payload to describe and provide the remote resources to reconnect to a user's disconnected sessions.</p> <p><a href="#">Click here to view this version of the [MS-RDWR] PDF.</a> ↗</p>
<a href="#">[MS-RMPR]: Rights Management Services (RMS): Client-to-Server Protocol</a>	<p>Specifies the Rights Management Services (RMS) Client-to-Server Protocol, a SOAP protocol used to obtain and issue certificates and licenses used for creating and working with protected content.</p> <p><a href="#">Click here to view this version of the [MS-RMPR] PDF.</a> ↗</p>
<a href="#">[MS-RMPPS]: Rights Management Services (RMS): Server-to-Server Protocol</a>	<p>Specifies the Rights Management Services (RMS): Server-to-Server Protocol, which is used to communicate information between RMS servers, implementing five interfaces, using either a binary-formatted interface over HTTP or a SOAP-based protocol over HTTP.</p> <p><a href="#">Click here to view this version of the [MS-RMPPS] PDF.</a> ↗</p>

Specification	Description
[MS-RMSI]: Rights Management Services (RMS): ISV Extension Protocol	<p>Specifies the Rights Management Services (RMS) ISV Extension Protocol, a SOAP protocol that is used to communicate information between applications and RMS servers directly without using the RMS client.</p> <p><a href="#">Click here to view this version of the [MS-RMSI] PDF.</a> ↗</p>
[MS-RNAP]: Vendor-Specific RADIUS Attributes for Network Access Protection (NAP) Data Structure	<p>Specifies the Vendor-Specific RADIUS Attributes for Network Access Protection (NAP) Data Structure protocol, which describes the Microsoft RADIUS vendor-specific attributes (VSAs) that are implemented in the Windows operating system.</p> <p><a href="#">Click here to view this version of the [MS-RNAP] PDF.</a> ↗</p>
[MS-RNAS]: Vendor-Specific RADIUS Attributes for Network Policy and Access Server Data Structure	<p>Specifies the Vendor-Specific RADIUS Attributes for the Network Policy and Access Server (NPAS) Data Structure protocol, which describes the Microsoft RADIUS vendor-specific attributes (VSAs) that are implemented in the Windows operating system.</p> <p><a href="#">Click here to view this version of the [MS-RNAS] PDF.</a> ↗</p>
[MS-RPCE]: Remote Procedure Call Protocol Extensions	<p>Specifies the Remote Procedure Call Protocol Extensions, a set of extensions to the DCE Remote Procedure Call 1.1 Specification, as specified in [C706]. These extensions add new capabilities to the DCE 1.1: RPC Specification, allow for more secure implementations to be built, and, in some cases, place additional restrictions on the DCE RPC Specification.</p> <p><a href="#">Click here to view this version of the [MS-RPCE] PDF.</a> ↗</p>
[MS-RPCH]: Remote Procedure Call over HTTP Protocol	<p>Specifies the Remote Procedure Call over HTTP Protocol, which describes the use of HTTP or HTTPS as a transport for the Remote Procedure Call (RPC) Protocol, as specified in [C706] and extended in [MS-RPCE].</p> <p><a href="#">Click here to view this version of the [MS-RPCH] PDF.</a> ↗</p>
[MS-RPCL]: Remote Procedure Call Location Services Extensions	<p>Specifies the Remote Procedure Call Location Services Extensions, a set of extensions and restrictions to the DCE Remote Procedure Call Location Services specification as defined in [C706].</p> <p><a href="#">Click here to view this version of the [MS-RPCL] PDF.</a> ↗</p>
[MS-RPRN]: Print System Remote Protocol	<p>Specifies the Print System Remote Protocol, which defines the communication of print job processing and print system management between a print client and a print server.</p> <p><a href="#">Click here to view this version of the [MS-RPRN] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-RRASM]: Routing and Remote Access Server (RRAS) Management Protocol</a>	<p>Specifies the Routing and Remote Access Server (RRAS) Management Protocol, which enables remote management (configuration and monitoring) of RRAS. The RRAS implementation refers to the components that can be configured to provide routing, remote access service, and site-to-site connectivity.</p> <p><a href="#">Click here to view this version of the [MS-RRASM] PDF.</a> ↗</p>
<a href="#">[MS-RRP]: Windows Remote Registry Protocol</a>	<p>Specifies the Windows Remote Registry Protocol, a remote procedure call (RPC)-based client/server protocol that is used to remotely manage a hierarchical data store such as the Windows registry.</p> <p><a href="#">Click here to view this version of the [MS-RRP] PDF.</a> ↗</p>
<a href="#">[MS-RRSP2]: Remote Rendering Server Protocol Version 2.0</a>	<p>Specifies the Remote Rendering Protocol Version 2, a user interface system for applications in Windows Media Center, which consists of an application-side component model connected to a remote renderer by an asynchronous messaging system that enables the quick and easy construction of captivating interfaces.</p> <p><a href="#">Click here to view this version of the [MS-RRSP2] PDF.</a> ↗</p>
<a href="#">[MS-RSMC]: Remote Session Monitoring and Control Protocol</a>	<p>Specifies and provides support for client machines to monitor and manage Remote Desktop Protocol (RDP) sessions on a server machine. The protocol provides a set of web service APIs that are implemented as a SOAP-based protocol that uses Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) as its transport.</p> <p><a href="#">Click here to view this version of the [MS-RSMC] PDF.</a> ↗</p>
<a href="#">[MS-RSMP]: Removable Storage Manager (RSM) Remote Protocol</a>	<p>Specifies the Removable Storage Manager (RSM) Remote Protocol, a set of distributed component object model (DCOM) interfaces for applications to manage robotic changers, media libraries, and tape drives. This protocol deals with detailed low-level operating system and storage concepts.</p> <p><a href="#">Click here to view this version of the [MS-RSMP] PDF.</a> ↗</p>
<a href="#">[MS-RSP]: Remote Shutdown Protocol</a>	<p>Specifies the Remote Shutdown Protocol, which is designed for shutting down, or for terminating the shutdown, of a remote computer during the shutdown waiting period.</p> <p><a href="#">Click here to view this version of the [MS-RSP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-RSVD]: Remote Shared Virtual Disk Protocol</a>	<p>Specifies the Remote Shared Virtual Disk Protocol, which supports accessing and manipulating virtual disks stored as files on an SMB3 file server. This protocol enables opening, querying, administering, reserving, reading, and writing the virtual disk objects, providing for flexible access by single or multiple consumers. It also provides for forwarding of SCSI operations, to be processed by the virtual disk.</p> <p><a href="#">Click here to view this version of the [MS-RSVD] PDF.</a></p>
<a href="#">[MS-RTPDT]: Real-Time Transport Protocol (RTP/RTCP): DTMF Digits, Telephony Tones and Telephony Signals Data Extensions</a>	<p>Specifies the Real-Time Transport Protocol (RTP/RTCP): DTMF Digits, Telephony Tones, and Telephony Signals Data Extensions, which describes the payload format needed to carry DTMF digits, tones, and signals in RTP packets over a network transport.</p> <p><a href="#">Click here to view this version of the [MS-RTPDT] PDF.</a></p>
<a href="#">[MS-RTPME]: Real-Time Transport Protocol (RTP/RTCP): Microsoft Extensions</a>	<p>Specifies the Real-Time Transport Protocol (RTP/RTCP): Microsoft Extensions, which is a set of network transport functions suitable for applications transmitting real-time data, such as audio and video, across multimedia endpoints.</p> <p><a href="#">Click here to view this version of the [MS-RTPME] PDF.</a></p>
<a href="#">[MS-RTPRAD]: Real-Time Transport Protocol (RTP/RTCP): Redundant Audio Data Extensions</a>	<p>Specifies the Real-Time Transport Protocol (RTP/RTCP): Redundant Audio Data Extensions, which encodes redundant audio data for use with the Real-Time Transport Protocol (RTP) Extensions protocol.</p> <p><a href="#">Click here to view this version of the [MS-RTPRAD] PDF.</a></p>
<a href="#">[MS-RTSP]: Real-Time Streaming Protocol (RTSP) Windows Media Extensions</a>	<p>Specifies the Real-Time Streaming Protocol (RTSP) Windows Media Extensions, which defines Windows Media extensions to the Real-Time Streaming Protocol (RTSP).</p> <p><a href="#">Click here to view this version of the [MS-RTSP] PDF.</a></p>
<a href="#">[MS-RXAD]: Remote Experience Advertisement Protocol</a>	<p>Specifies the Remote Experience Advertisement Protocol, which enables a Universal Plug and Play (UPnP) service implemented by a device to be used by the client to advertise available remote experience information to that device.</p> <p><a href="#">Click here to view this version of the [MS-RXAD] PDF.</a></p>

Specification	Description
<a href="#">[MS-SAMLPR]: Security Assertion Markup Language (SAML) Proxy Request Signing Protocol</a>	<p>Specifies the Security Assertion Markup Language (SAML) Proxy Request Signing Protocol, which allows proxy servers to perform operations that require knowledge of configured keys and other state information about federated sites known by the Security Token service server.</p> <p><a href="#">Click here to view this version of the [MS-SAMLPR] PDF.</a></p>
<a href="#">[MS-SAMR]: Security Account Manager (SAM) Remote Protocol (Client-to-Server)</a>	<p>Specifies the Security Account Manager (SAM) Remote Protocol, which supports management functionality for an account store or directory containing users and groups. The goal of the protocol is to enable IT administrators and users to manage users, groups, and computers.</p> <p><a href="#">Click here to view this version of the [MS-SAMR] PDF.</a></p>
<a href="#">[MS-SAMS]: Security Account Manager (SAM) Remote Protocol (Server-to-Server)</a>	<p>Specifies the Security Account Manager (SAM) Remote Protocol (Server-to-Server). Domain controllers (DCs) use this protocol to forward time-critical database changes to the primary domain controller (PDC), and to forward time-critical database changes from a read-only domain controller (RODC) to a writable NC replica within the same domain outside the normal replication protocol.</p> <p><a href="#">Click here to view this version of the [MS-SAMS] PDF.</a></p>
<a href="#">[MS-SCMP]: Shadow Copy Management Protocol</a>	<p>Specifies the Shadow Copy Management Protocol, which programmatically enumerates shadow copies and configures shadow copy storage on remote machines.</p> <p><a href="#">Click here to view this version of the [MS-SCMP] PDF.</a></p>
<a href="#">[MS-SCMR]: Service Control Manager Remote Protocol</a>	<p>Specifies the Service Control Manager Remote Protocol, which is used for remotely managing the Service Control Manager (SCM), an RPC server that enables service configuration and control of service programs.</p> <p><a href="#">Click here to view this version of the [MS-SCMR] PDF.</a></p>
<a href="#">[MS-SDP]: Session Description Protocol (SDP) Extensions</a>	<p>Specifies the Session Description Protocol (SDP) Extensions, which describes the session description that is used to negotiate instant messaging, audio and video, and data collaboration sessions, and notes the extensions used.</p> <p><a href="#">Click here to view this version of the [MS-SDP] PDF.</a></p>

Specification	Description
<a href="#">[MS-SFMWA]: Server and File Management Web APIs Protocol</a>	<p>Specifies the Server and File Management Web APIs Protocol, which is used to access a REST-based server and to manage files over the HTTPS transports. The protocol exposes a set of built-in web services for third-party developers to build applications on different devices that can access files and manage servers remotely. The protocol also allows third-party developers to add their own web services without the need to handle authentication.</p> <p><a href="#">Click here to view this version of the [MS-SFMWA] PDF.</a> ↗</p>
<a href="#">[MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol</a>	<p>Specifies the Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol, which are two extensions to the Kerberos protocol as developed by Microsoft. These two extensions, collectively known as Service for User (S4U), enable an application service to obtain a Kerberos service ticket on behalf of a user.</p> <p><a href="#">Click here to view this version of the [MS-SFU] PDF.</a> ↗</p>
<a href="#">[MS-SHLLINK]: Shell Link (.LNK) Binary File Format</a>	<p>Specifies the Shell Link Binary File Format, which contains information that can be used to access another data object. The Shell Link Binary File Format is the format of Windows files with the extension "LNK".</p> <p><a href="#">Click here to view this version of the [MS-SHLLINK] PDF.</a> ↗</p>
<a href="#">[MS-SIP]: Session Initiation Protocol Extensions</a>	<p>Specifies Microsoft extensions to the Session Initiation Protocol (SIP), as specified in [RFC3261], which is used by terminals to establish, modify, and terminate multimedia sessions or calls. The SIP extensions add support for privacy features and for subscription requests for offline end nodes to the SIP extensions for presence.</p> <p><a href="#">Click here to view this version of the [MS-SIP] PDF.</a> ↗</p>
<a href="#">[MS-SMB]: Server Message Block (SMB) Protocol</a>	<p>Specifies the Server Message Block (SMB) Protocol, which defines extensions to the existing Common Internet File System (CIFS) specification that have been implemented by Microsoft since the publication of the [CIFS] specification.</p> <p><a href="#">Click here to view this version of the [MS-SMB] PDF.</a> ↗</p>
<a href="#">[MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3</a>	<p>Specifies the Server Message Block (SMB) Protocol Versions 2 and 3, which support the sharing of file and print resources between machines and extend the concepts from the Server Message Block Protocol.</p> <p><a href="#">Click here to view this version of the [MS-SMB2] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-SMBD]: SMB2 Remote Direct Memory Access (RDMA) Transport Protocol</a>	<p>Specifies the SMB2 Remote Direct Memory Access (RDMA) Transport Protocol, a wrapper for the existing SMB2 protocol that allows SMB2 packets to be delivered over RDMA-capable transports such as iWARP or Infiniband while utilizing the direct data placement (DDP) capabilities of these transports. Benefits include reduced CPU overhead, lower latency, and improved throughput.</p> <p><a href="#">Click here to view this version of the [MS-SMBD] PDF.</a></p>
<a href="#">[MS-SMTPNTLM]: NT LAN Manager (NTLM) Authentication: Simple Mail Transfer Protocol (SMTP) Extension</a>	<p>Specifies the NT LAN Manager (NTLM) Authentication: Simple Mail Transfer Protocol (SMTP) Extension, which uses NT LAN Manager (NTLM) authentication (as specified in [MS-NLMP]) by the Simple Mail Transfer Protocol (SMTP) to facilitate client authentication to a Windows SMTP server.</p> <p><a href="#">Click here to view this version of the [MS-SMTPNTLM] PDF.</a></p>
<a href="#">[MS-SNID]: Server Network Information Discovery Protocol</a>	<p>Specifies the Server Network Information Discovery Protocol, which defines a pair of request and response messages by which a protocol client can locate protocol servers within the broadcast/multicast scope. The client can then get network information (such as NetBIOS name, Internet Protocol version 4 (IPv4), and Internet Protocol version 6 (IPv6) addresses) about the servers.</p> <p><a href="#">Click here to view this version of the [MS-SNID] PDF.</a></p>
<a href="#">[MS-SNTP]: Network Time Protocol (NTP) Authentication Extensions</a>	<p>Specifies the Network Time Protocol (NTP) Authentication Extensions, which is an authentication extension to the Network Time Protocol (NTP) version 3 ([RFC1305]) and the Simple Network Time Protocol (SNTP) version 4 ([RFC2030]).</p> <p><a href="#">Click here to view this version of the [MS-SNTP] PDF.</a></p>
<a href="#">[MS-SPNG]: Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension</a>	<p>Specifies the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Protocol Extension. SPNEGO is a security protocol that uses a GSS-API authentication mechanism. GSS-API is a literal set of functions that include both an API and a methodology for approaching authentication.</p> <p><a href="#">Click here to view this version of the [MS-SPNG] PDF.</a></p>
<a href="#">[MS-SQMCS]: Software Quality Metrics (SQM) Client-to-Service Version 1 Protocol</a>	<p>Specifies the Software Quality Metrics (SQM) Client-to-Service Protocol V1, used to send software instrumentation metrics to the SQM service and by the client to download client-specific control data. The protocol allows applications and operating system components to collect and send instrumentation metrics to a hosted service.</p> <p><a href="#">Click here to view this version of the [MS-SQMCS] PDF.</a></p>

Specification	Description
<a href="#">[MS-SQMCS2]: Software Quality Metrics (SQM) Client-to-Service Version 2 Protocol</a>	<p>Specifies the Software Quality Metrics (SQM) Client-to-Service Protocol V2, which is used to send software instrumentation metrics to the SQM service and for the client to download client-specific control data. The protocol extends the concepts of the Software Quality Metrics (SQM) Client-to-Service Protocol, as specified in [MS-SQMCS].</p> <p><a href="#">Click here to view this version of the [MS-SQMCS2] PDF.</a> ↗</p>
<a href="#">[MS-SQOS]: Storage Quality of Service Protocol</a>	<p>Specifies the Storage Quality of Service (QoS) Protocol, which is a block-based protocol that is used to manage the Quality of Service configuration of I/O flows targeting remote files accessed over SMB3.</p> <p><a href="#">Click here to view this version of the [MS-SQOS] PDF.</a> ↗</p>
<a href="#">[MS-SRPL]: Directory Replication Service (DRS) Protocol Extensions for SMTP</a>	<p>Specifies the Directory Replication Service (DRS) Protocol Extensions for SMTP. These are extensions to the DRS Protocol for transport over the Simple Mail Transfer Protocol (SMTP), which provide an alternate transport for the DRS protocol that may allow domain controllers to perform replication in environments where the RPC transport mechanism is unsuitable.</p> <p><a href="#">Click here to view this version of the [MS-SRPL] PDF.</a> ↗</p>
<a href="#">[MS-SRVS]: Server Service Remote Protocol</a>	<p>Specifies the Server Service Remote Protocol, which remotely enables file and printer sharing and named pipe access to the server through the Server Message Block Protocol.</p> <p><a href="#">Click here to view this version of the [MS-SRVS] PDF.</a> ↗</p>
<a href="#">[MS-SSDP]: SSDP: Networked Home Entertainment Devices (NHED) Extensions</a>	<p>Specifies the Networked Home Entertainment Devices (NHED) Extensions, which detects devices on a home network. These extensions provide a mechanism for a control point to discover a device on the network without requiring the device to implement a complete SSDP stack.</p> <p><a href="#">Click here to view this version of the [MS-SSDP] PDF.</a> ↗</p>
<a href="#">[MS-SSEAN]: Simple Mail Transfer Protocol (SMTP) AUTH Extension for SPNEGO</a>	<p>Specifies the SMTP Service Extension for Negotiate Authentication, which enables SMTP clients to authenticate to SMTP servers by using the Simple and Protected Negotiate (SPNEGO) mechanism.</p> <p><a href="#">Click here to view this version of the [MS-SSEAN] PDF.</a> ↗</p>
<a href="#">[MS-SSTP]: Secure Socket Tunneling Protocol (SSTP)</a>	<p>Specifies the Secure Socket Tunneling Protocol (SSTP), which is a mechanism to transport data-link layer (L2) frames on a Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) connection.</p> <p><a href="#">Click here to view this version of the [MS-SSTP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-SSTR]: Smooth Streaming Protocol</a>	<p>Specifies the Smooth Streaming Protocol, which provides a means of delivering media from servers to clients in a way that can be cached by standard HTTP Cache Proxies in the communication chain. Allowing standard HTTP Cache Proxies to respond to requests on behalf of the server increases the number of clients that can be served by a single server.</p> <p><a href="#">Click here to view this version of the [MS-SSTR] PDF.</a></p>
<a href="#">[MS-SWN]: Service Witness Protocol</a>	<p>Specifies the Service Witness Protocol, which enables an SMB2 clustered file server to notify SMB2 clients with prompt and explicit notifications about the failure or recovery of a network name and associated services.</p> <p><a href="#">Click here to view this version of the [MS-SWN] PDF.</a></p>
<a href="#">[MS-SWSB]: SOAP Over WebSocket Protocol Binding</a>	<p>Specifies the SOAP over WebSocket Protocol Binding, a binding of SOAP to the WebSocket protocol (as defined in [DRAFT-WSP]), including a WSDL transport URI and supported message exchange patterns (MEPs). It specifies how messages defined by a higher-layer protocol are formed and framed for transport over [DRAFT-WSP]. This specification also defines a WebSocket subprotocol.</p> <p><a href="#">Click here to view this version of the [MS-SWSB] PDF.</a></p>
<a href="#">[MS-TAIL]: Telephony API Internet Locator Service Protocol</a>	<p>Specifies the Telephony API Internet Locator Service Protocol, which uses Lightweight Directory Access Protocol (LDAP) requests to retrieve information stored in the Internet Locator Service (ILS) dynamic instance. It is used for communication between a client using the Telephony Application Programming Interface (TAPI) and an ILS server.</p> <p><a href="#">Click here to view this version of the [MS-TAIL] PDF.</a></p>
<a href="#">[MS-TCC]: Tethering Control Channel Protocol</a>	<p>Specifies the Tethering Control Channel Protocol, which enables the sharing of the network connection for a server with one or more clients.</p> <p><a href="#">Click here to view this version of the [MS-TCC] PDF.</a></p>
<a href="#">[MS-TDS]: Tabular Data Stream Protocol</a>	<p>Specifies the Tabular Data Stream Protocol, which is an application layer request/response protocol that facilitates interaction with a database server and provides for authentication and channel encryption negotiation; specification of requests in SQL (including Bulk Insert); invocation of a stored procedure, also known as a Remote Procedure Call (RPC); returning of data; and Transaction Manager Requests.</p> <p><a href="#">Click here to view this version of the [MS-TDS] PDF.</a></p>
<a href="#">[MS-THCH]: Tracing HTTP Correlation Header Protocol</a>	<p>Specifies the Tracing HTTP Correlation Header, which is used to enable correlation between client and server-side traces.</p> <p><a href="#">Click here to view this version of the [MS-THCH] PDF.</a></p>

Specification	Description
<a href="#">[MS-TIPP]: Transaction Internet Protocol (TIP) Extensions</a>	<p>Specifies the Transaction Internet Protocol (TIP) Extensions, which is a set of extensions to the standard Transaction Internet Protocol (TIP) Version 3.0, as specified in [RFC2371]. The protocol provides concrete mechanisms for associating an OleTx transaction and a TIP transaction.</p> <p><a href="#">Click here to view this version of the [MS-TIPP] PDF.</a> ↗</p>
<a href="#">[MS-TLSP]: Transport Layer Security (TLS) Profile</a>	<p>Specifies the Transport Layer Security (TLS) Profile, which is the authentication option to the Telnet protocol as a generic method for negotiating an authentication type and mode, including determining whether encryption should be used and whether credentials should be forwarded.</p> <p><a href="#">Click here to view this version of the [MS-TLSP] PDF.</a> ↗</p>
<a href="#">[MS-TNAP]: Telnet: NT LAN Manager (NTLM) Authentication Protocol</a>	<p>Specifies the Telnet: NT LAN Manager (NTLM) Authentication Protocol, which is the authentication option to the Telnet protocol as a generic method for negotiating an authentication type and mode, including determining whether encryption should be used and whether credentials should be forwarded.</p> <p><a href="#">Click here to view this version of the [MS-TNAP] PDF.</a> ↗</p>
<a href="#">[MS-TPMVSC]: Trusted Platform Module (TPM) Virtual Smart Card Management Protocol</a>	<p>Specifies the DCOM Interfaces for Trusted Platform Module (TPM) Virtual Smart Card device management, which are used to manage virtual smart cards (VSCs) on a remote machine. They provide methods for a protocol client to request creation and destruction of VSCs, and to monitor the status of these operations.</p> <p><a href="#">Click here to view this version of the [MS-TPMVSC] PDF.</a> ↗</p>
<a href="#">[MS-TPXS]: Telemetry Protocol XML Schema</a>	<p>Specifies the Telemetry Protocol XML Schema. This schema defines the message structure used by the Software Quality Metrics (SQM) Client-to-Service Protocol V2, specified in [MS-SQMCS2]. The schema is used to send software instrumentation metrics from a client to the SQM service and for the client to download client-specific control data.</p> <p><a href="#">Click here to view this version of the [MS-TPXS] PDF.</a> ↗</p>
<a href="#">[MS-TRP]: Telephony Remote Protocol</a>	<p>Specifies the Telephony Remote Protocol, which enables implementation of communications applications ranging from voice mail to call centers with multiple agents and switches.</p> <p><a href="#">Click here to view this version of the [MS-TRP] PDF.</a> ↗</p>
<a href="#">[MS-TSCH]: Task Scheduler Service Remoting Protocol</a>	<p>Specifies the Task Scheduler Service Remoting Protocol, which is used to register and configure a task and to inquire about the status of tasks that are running on a remote machine.</p> <p><a href="#">Click here to view this version of the [MS-TSCH] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-TSGU]: Terminal Services Gateway Server Protocol</a>	<p>Specifies the Terminal Services Gateway Server Protocol, which is a mechanism to transport data-link layer (L2) frames on a Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) connection.</p> <p><a href="#">Click here to view this version of the [MS-TSGU] PDF.</a> ↗</p>
<a href="#">[MS-TSRAP]: Telnet Server Remote Administration Protocol</a>	<p>Specifies the Telnet Server Remote Administration Protocol, which is a set of interfaces used for performing management tasks on a Telnet Server.</p> <p><a href="#">Click here to view this version of the [MS-TSRAP] PDF.</a> ↗</p>
<a href="#">[MS-TSTS]: Terminal Services Terminal Server Runtime Interface Protocol</a>	<p>Specifies the Terminal Services Terminal Server Runtime Interface Protocol, which is an RPC-based protocol used for remotely querying and configuring various aspects of a terminal server.</p> <p><a href="#">Click here to view this version of the [MS-TSTS] PDF.</a> ↗</p>
<a href="#">[MS-TSWP]: Terminal Services Workspace Provisioning Protocol</a>	<p>Specifies the Terminal Services Workspace Provisioning Protocol, which is used for transferring remote resource information from a server to a client. The client can use this resource information to launch resources such as remote applications on a remote server.</p> <p><a href="#">Click here to view this version of the [MS-TSWP] PDF.</a> ↗</p>
<a href="#">[MS-TVTT]: Telnet: VTNT Terminal Type Format Data Structure</a>	<p>Specifies the Telnet: VTNT Terminal Type Format Data Structure, which defines the structures for Telnet VTNT Terminal Type Format, and how the client and server negotiate the use of this format.</p> <p><a href="#">Click here to view this version of the [MS-TVTT] PDF.</a> ↗</p>
<a href="#">[MS-UAMG]: Update Agent Management Protocol</a>	<p>Specifies the Update Agent Management Protocol, which provides a set of types and interfaces that allows callers to manage an update agent and to invoke some update agent operations, such as an update search.</p> <p><a href="#">Click here to view this version of the [MS-UAMG] PDF.</a> ↗</p>
<a href="#">[MS-UNMP]: User Name Mapping Protocol</a>	<p>Specifies the User Name Mapping Protocol, which maps Windows domain user and group account names to the POSIX user and group identifiers used in AUTH_UNIX authentication, and vice versa. This enables the association of user names for users who have different identities in Windows-based and UNIX-based domains.</p> <p><a href="#">Click here to view this version of the [MS-UNMP] PDF.</a> ↗</p>

Specification	Description
[MS-UPIGD]: UPnP Device and Service Templates: Internet Gateway Device (IGD) Extensions	<p>Specifies the UPnP: Device and Service Templates: Internet Gateway Device (IGD) Extensions. These structure extensions define extensions to the Universal Plug-n-Play (UPnP) device schema that describes an Internet gateway device.</p> <p><a href="#">Click here to view this version of the [MS-UPIGD] PDF.</a> ↗</p>
[MS-UPMC]: UPnP Device and Service Templates: Media Property and Compatibility Extensions	<p>Specifies the Microsoft Media Property Extensions (MMPE), the Microsoft Compatibility Extension Flags (MCEF), and the Microsoft Power Management Extensions (MPME) to the Universal Plug and Play (UPnP) interoperability guidelines, as specified by the UPnP Forum [UPnP] and used by the Digital Living Network Alliance (DLNA) [DLNA].</p> <p><a href="#">Click here to view this version of the [MS-UPMC] PDF.</a> ↗</p>
[MS-V4OF]: IPv4 Over IEEE 1394 Protocol Extensions	<p>Specifies the IPv4 Over IEEE 1394 Protocol Extension, which is the Microsoft extension to the IPv4 over IEEE 1394 protocol to support bridging and clarifies the implementation details as specified in [RFC2734] where necessary.</p> <p><a href="#">Click here to view this version of the [MS-V4OF] PDF.</a> ↗</p>
[MS-VAPR]: Virtual Application Publishing and Reporting (App-V) Protocol	<p>Specifies the virtual applications that a user is entitled to so that these applications can be downloaded and installed on the user's machine. It is also used to report virtual application usage information to the server so that usage information across multiple users can be aggregated to infer broad virtual application usage patterns across an organization.</p> <p><a href="#">Click here to view this version of the [MS-VAPR] PDF.</a> ↗</p>
[MS-VDS]: Virtual Disk Service (VDS) Protocol	<p>Specifies the Virtual Disk Service (VDS) Protocol, a set of distributed component object model (DCOM) interfaces for managing the configuration of disk storage.</p> <p><a href="#">Click here to view this version of the [MS-VDS] PDF.</a> ↗</p>
[MS-VHDX]: Virtual Hard Disk v2 (VHDX) File Format	<p>Specifies the Virtual Hard Disk v2 (VHDX) File Format Protocol, the virtual hard disk format that provides a disk-in-a-file abstraction.</p> <p><a href="#">Click here to view this version of the [MS-VHDX] PDF.</a> ↗</p>
[MS-VUVP]: VT-UTF8 and VT100+ Protocols	<p>Specifies the VT-UTF8 and VT100+ Protocols, which are used for point-to-point serial communication for terminal control and headless server configuration.</p> <p><a href="#">Click here to view this version of the [MS-VUVP] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-W32T]: W32Time Remote Protocol</a>	<p>Specifies the W32Time Remote Protocol, which is used for controlling and monitoring a time service on a machine. This RPC interface supports time services that synchronize time using the Network Time Protocol (NTP) Version 3, as specified in [RFC1305], as well as platform-specific hardware time sources.</p> <p><a href="#">Click here to view this version of the [MS-W32T] PDF.</a></p>
<a href="#">[MS-WCCE]: Windows Client Certificate Enrollment Protocol</a>	<p>Specifies the Windows Client Certificate Enrollment Protocol, which consists of a set of DCOM interfaces that enable clients to request various services from a certification authority (CA). These services enable X.509 (as specified in [X509]) digital certificate enrollment, issuance, revocation, and property retrieval.</p> <p><a href="#">Click here to view this version of the [MS-WCCE] PDF.</a></p>
<a href="#">[MS-WCFESAN]: WCF-Based Encrypted Server Administration and Notification Protocol</a>	<p>Specifies the WCF-Based Encrypted Server Administration and Notification Protocol, which enables the protocol client to monitor and manage the protocol server in the same network.</p> <p><a href="#">Click here to view this version of the [MS-WCFESAN] PDF.</a></p>
<a href="#">[MS-WDHCE]: Wi-Fi Display Protocol: Hardware Cursor Extension</a>	<p>Specifies the Wi-Fi Display Protocol: Hardware Cursor Extension, which extends the Miracast v1.1 protocol to provide an additional, low-latency stream suitable for controlling the mouse cursor at a higher update rate.</p> <p><a href="#">Click here to view this version of the [MS-WDHCE] PDF.</a></p>
<a href="#">[MS-WDSC]: Windows Deployment Services Control Protocol</a>	<p>Specifies the Windows Deployment Services (WDS) Control Protocol, which is an RPC interface that provides the ability to remotely invoke services provided by WDS Server. It is a client/server protocol that uses RPC as a transport. The protocol provides a generic invocation mechanism to send requests to the server and receive replies.</p> <p><a href="#">Click here to view this version of the [MS-WDSC] PDF.</a></p>
<a href="#">[MS-WDSMA]: Windows Deployment Services Multicast Application Protocol</a>	<p>Specifies the Windows Deployment Services Multicast Application Protocol, which enables clients to join the multicast session at any point during the lifetime of the .multicast session, and still be able to get all pieces of the content.</p> <p><a href="#">Click here to view this version of the [MS-WDSMA] PDF.</a></p>

Specification	Description
<a href="#">[MS-WDSMSI]: Windows Deployment Services Multicast Session Initiation Protocol</a>	<p>Specifies the Windows Deployment Services Multicast Session Initiation Protocol, which describes two mechanisms for the client to request initiation of a Multicast Session from the server.</p> <p><a href="#">Click here to view this version of the [MS-WDSMSI] PDF.</a></p>
<a href="#">[MS-WDSMT]: Windows Deployment Services Multicast Transport Protocol</a>	<p>Specifies the Windows Deployment Services Multicast Transport Protocol, which enables transmission of content to multiple clients using Multicast UDP.</p> <p><a href="#">Click here to view this version of the [MS-WDSMT] PDF.</a></p>
<a href="#">[MS-WDSOSD]: Windows Deployment Services Operation System Deployment Protocol</a>	<p>Specifies the Windows Deployment Services Operation System Deployment Protocol . This protocol defines services exposed by the WDS Server that are used by the clients to deploy an operating system on a machine.</p> <p><a href="#">Click here to view this version of the [MS-WDSOSD] PDF.</a></p>
<a href="#">[MS-WDV]: Web Distributed Authoring and Versioning (WebDAV) Protocol: Client Extensions</a>	<p>Specifies the Web Distributed Authoring and Versioning (WebDAV) Protocol: Client Extensions, which extends WebDAV by introducing new headers that both enable the file types that are not currently manageable and optimize protocol interactions for file system clients. These extensions do not introduce new functionality into WebDAV, but instead optimize processing and eliminate the need for special-case processing.</p> <p><a href="#">Click here to view this version of the [MS-WDV] PDF.</a></p>
<a href="#">[MS-WDVSE]: Web Distributed Authoring and Versioning (WebDAV) Protocol: Server Extensions</a>	<p>Specifies the Web Distributed Authoring and Versioning (WebDAV) Protocol: Server Extension, which extends the standard HTTP mechanisms defined in [RFC2068] to provide file access and content management over the Internet.</p> <p><a href="#">Click here to view this version of the [MS-WDVSE] PDF.</a></p>
<a href="#">[MS-WFDDAA]: Wi-Fi Direct (WFD) Application to Application Protocol</a>	<p>Specifies the Wi-Fi Direct (WFD) Protocol: Proximity Extensions, which enable two or more devices that are running the same application to establish a direct connection without requiring an intermediary, such as an infrastructure wireless access point (WAP).</p> <p><a href="#">Click here to view this version of the [MS-WFDDAA] PDF.</a></p>

Specification	Description
<a href="#">[MS-WFDPE]: Wi-Fi Display Protocol Extension</a>	<p>Specifies an extension for the Wi-Fi Display Technical Specification v1.1. Enables latency control, extended diagnostic information, and dynamic format changes on Wi-Fi Display Devices. When implemented, these extensions provide an improved and more consistent Wi-Fi Display experience for a variety of wireless display scenarios, including word processing, web browsing, gaming, and video projection.</p> <p><a href="#">Click here to view this version of the [MS-WFDPE] PDF.</a></p>
<a href="#">[MS-WFIM]: Workflow Instance Management Protocol</a>	<p>Specifies the Workflow Instance Management Protocol, which defines a set of SOAP messages for the management of workflow instances, such as suspending, resuming, or canceling an instance.</p> <p><a href="#">Click here to view this version of the [MS-WFIM] PDF.</a></p>
<a href="#">[MS-WINSRA]: Windows Internet Naming Service (WINS) Replication and Autodiscovery Protocol</a>	<p>Specifies the Windows Internet Naming Service (WINS) Replication and Autodiscovery Protocol, the Microsoft implementation of NetBIOS Name Server (NBNS). This protocol supports resolution of NetBIOS names to IPv4 addresses.</p> <p><a href="#">Click here to view this version of the [MS-WINSRA] PDF.</a></p>
<a href="#">[MS-WKST]: Workstation Service Remote Protocol</a>	<p>Specifies the Workstation Service Remote Protocol, which remotely queries and configures certain aspects of a Server Message Block network redirector on a remote computer.</p> <p><a href="#">Click here to view this version of the [MS-WKST] PDF.</a></p>
<a href="#">[MS-WMF]: Windows Metafile Format</a>	<p>Specifies the Windows Metafile Format structure. A Windows metafile is a container for an image, which is defined by series of variable-length records, called metafile records.</p> <p><a href="#">Click here to view this version of the [MS-WMF] PDF.</a></p>
<a href="#">[MS-WMHTTP]: Windows Media HTTP Push Distribution Protocol</a>	<p>Specifies the Windows Media HTTP Push Distribution Protocol, which is used for transferring real-time multimedia data (for example, audio and video) from a client to a server.</p> <p><a href="#">Click here to view this version of the [MS-WMHTTP] PDF.</a></p>
<a href="#">[MS-WMI]: Windows Management Instrumentation Remote Protocol</a>	<p>Specifies the Windows Management Instrumentation Remote Protocol, which uses the Common Information Model (CIM), as specified in [DMTF-DSP004], to represent various components of the operating system. CIM is the conceptual model for storing enterprise management information.</p> <p><a href="#">Click here to view this version of the [MS-WMI] PDF.</a></p>

Specification	Description
<a href="#">[MS-WMIO]: Windows Management Instrumentation Encoding Version 1.0 Protocol</a>	<p>Specifies the Windows Management Instrumentation Encoding Version 1.0 Protocol, which is a binary data encoding format used by the Windows Management Instrumentation Remote Protocol, as specified in [MS-WMI], for network communication.</p> <p><a href="#">Click here to view this version of the [MS-WMIO] PDF.</a> ↗</p>
<a href="#">[MS-WMLOG]: Windows Media Log Data Structure</a>	<p>Specifies the Windows Media Log Data Structure, which is a syntax for logging messages. The logging messages specify information about how a client received multimedia content from a streaming server.</p> <p><a href="#">Click here to view this version of the [MS-WMLOG] PDF.</a> ↗</p>
<a href="#">[MS-WMSP]: Windows Media HTTP Streaming Protocol</a>	<p>Specifies the Windows Media HTTP Streaming Protocol, a client/server-based protocol used to stream real-time data between the client (the receiver of streaming data) and server (the sender of streaming data).</p> <p><a href="#">Click here to view this version of the [MS-WMSP] PDF.</a> ↗</p>
<a href="#">[MS-WPRN]: Web Point-and-Print Protocol</a>	<p>Specifies the Web Point-and-Print Protocol, which is an HTTP-based protocol that clients use to download printer driver software from a server in the client network or from a Web site. This enables distribution of printer driver software using standard Web technologies.</p> <p><a href="#">Click here to view this version of the [MS-WPRN] PDF.</a> ↗</p>
<a href="#">[MS-WSDS]: WS-Enumeration: Directory Services Protocol Extensions</a>	<p>Specifies the WS-Enumeration Directory Services Protocol Extensions, a set of extensions to the Web Services Enumeration (WS-Enumeration) [WSENUM] protocol for facilitating SOAP-based search operations against directory servers.</p> <p><a href="#">Click here to view this version of the [MS-WSDS] PDF.</a> ↗</p>
<a href="#">[MS-WSH]: Windows Security Health Agent (WSHA) and Windows Security Health Validator (WSHV) Protocol</a>	<p>Specifies the Windows Security Health Agent (WSHA) and Windows Security Health Validator (WSHV) Protocol, which reports the system security health state.</p> <p><a href="#">Click here to view this version of the [MS-WSH] PDF.</a> ↗</p>
<a href="#">[MS-WSMAN]: Web Services Management Protocol Extensions for Windows Server 2003</a>	<p>Specifies the Web Services Management Protocol Extensions, which is a general purpose, SOAP-based systems management extension that defines procedures for carrying out remote management operations.</p> <p><a href="#">Click here to view this version of the [MS-WSMAN] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-WSMV]: Web Services Management Protocol Extensions for Windows Vista</a>	<p>Specifies the Web Services Management Protocol Extensions for Windows Vista, which provides Windows Vista extensions to the WS-Management Protocol, the WS-Management Binding Specification, and the WS-CIM Mapping Specification for accessing CIM objects as a Web service.</p> <p><a href="#">Click here to view this version of the [MS-WSMV] PDF.</a></p>
<a href="#">[MS-WSP]: Windows Search Protocol</a>	<p>Specifies the Windows Search Protocol (WSP), which allows a client to communicate with a server hosting a Windows Search service (WSS) to issue queries.</p> <p><a href="#">Click here to view this version of the [MS-WSP] PDF.</a></p>
<a href="#">[MS-WSPE]: WebSocket Protocol Extensions</a>	<p>Specifies the WebSocket Protocol: Disable Masking Extension, which extends the WebSocket Protocol to improve performance by allowing developers to set a property to disable masking.</p> <p><a href="#">Click here to view this version of the [MS-WSPE] PDF.</a></p>
<a href="#">[MS-WSPELD]: WS-Transfer and WS-Enumeration Protocol Extension for Lightweight Directory Access Protocol v3 Controls</a>	<p>Specifies the WS-Transfer: Lightweight Directory Access Protocol (LDAP) v3 Controls, also known as WSPELD. This protocol extends the Web Services Enumeration (WS-Enumeration) [WSENUM] and Web Services Transfer (WS-Transfer) [WXFR] protocols.</p> <p><a href="#">Click here to view this version of the [MS-WSPELD] PDF.</a></p>
<a href="#">[MS-WSPOL]: Web Services: Policy Assertions and WSDL Extensions</a>	<p>Specifies a collection of Web service policy assertions, which define domain-specific behavior for the interaction between two Web service entities.</p> <p><a href="#">Click here to view this version of the [MS-WSPOL] PDF.</a></p>
<a href="#">[MS-WSRM]: Windows System Resource Manager (WSRM) Protocol</a>	<p>Specifies the Windows System Resource Manager (WSRM) Protocol, a set of Distributed Component Object Model (DCOM) interfaces for managing the configuration of processor, memory resources, and accounting functions on a server.</p> <p><a href="#">Click here to view this version of the [MS-WSRM] PDF.</a></p>
<a href="#">[MS-WSRVCAT]: WS-AtomicTransaction (WS-AT) Version 1.0 Protocol Extensions</a>	<p>Specifies the WS-AtomicTransaction (WS-AT) Version 1.0 Protocol Extensions, which extends the WS-AtomicTransaction protocol by enabling WS-AtomicTransaction initiators, participants, and coordinators to participate in transactions coordinated by OleTx transaction managers.</p> <p><a href="#">Click here to view this version of the [MS-WSRVCAT] PDF.</a></p>

Specification	Description
[MS-WSRVCRM]: WS-ReliableMessaging Protocol: Advanced Flow Control Extension	<p>Specifies the WS-ReliableMessaging Protocol: Advanced Flow Control Extension, which is an advanced message flow control extension to the Web Services Reliable Messaging Protocol [WSRM1-0] [WSRM1-1].</p> <p><a href="#">Click here to view this version of the [MS-WSRVCRM] PDF.</a> ↗</p>
[MS-WSRVCRR]: WS-ReliableMessaging Protocol: Reliable Request-Reply Extension	<p>Specifies the WS-ReliableMessaging Protocol: Reliable Request-Reply Extension. This extension assumes the use of duplex underlying protocols in order to provide support for applications designed to interact using a request-response message exchange pattern. The request-reply extension enables these applications to communicate reliably over transfer protocols that support only SOAP Request-Response.</p> <p><a href="#">Click here to view this version of the [MS-WSRVCRR] PDF.</a> ↗</p>
[MS-WSTC]: WS-Discovery: Termination Criteria Protocol Extensions	<p>Specifies the WS-Discovery: Termination Criteria Protocol Extensions. This extends the WS-Discovery protocol for sending and receiving termination criteria as part of WS-Discovery Probe and Resolve messages.</p> <p><a href="#">Click here to view this version of the [MS-WSTC] PDF.</a> ↗</p>
[MS-WSTEP]: WS-Trust X.509v3 Token Enrollment Extensions	<p>Specifies the WS-Trust X.509v3 Token Enrollment Extensions, also known as WSTEP. The protocol specification defines the message formats and server behavior for the purposes of certificate enrollment.</p> <p><a href="#">Click here to view this version of the [MS-WSTEP] PDF.</a> ↗</p>
[MS-WTIM]: WS-Transfer: Identity Management Operations for Directory Access Extensions	<p>Specifies the WS-Transfer: Identity Management Operations for Directory Access Extensions, a set of extensions to the WS-Transfer protocol [WXFR] for representing the protocol operations commonly used for directory access in identity management protocols.</p> <p><a href="#">Click here to view this version of the [MS-WTIM] PDF.</a> ↗</p>
[MS-WSUSAR]: Windows Server Update Services: Administrative API Remoting Protocol (WSUSAR)	<p>Specifies the Windows Server Update Services: Administrative API Remoting Protocol (WSUSAR), which enables communication between the Windows Server Update Services (WSUS) management API and a WSUS server.</p> <p><a href="#">Click here to view this version of the [MS-WSUSAR] PDF.</a> ↗</p>
[MS-WSUSSS]: Windows Update Services: Server-Server Protocol	<p>Specifies the Windows Update Services: Server-Server Protocol, which enables a hierarchically organized collection of servers to synchronize metadata and content associated with software updates over the Internet by using SOAP and HTTP protocols.</p> <p><a href="#">Click here to view this version of the [MS-WSUSSS] PDF.</a> ↗</p>

Specification	Description
<a href="#">[MS-WUSP]: Windows Update Services: Client-Server Protocol</a>	<p>Specifies the Windows Update Services: Client-Server Protocol, which enables machines to discover and download software updates over the Internet using the SOAP and HTTP protocols.</p> <p><a href="#">Click here to view this version of the [MS-WUSP] PDF.</a></p>
<a href="#">[MS-XCA]: Xpress Compression Algorithm</a>	<p>Specifies the three variants of the Xpress Compression Algorithm: LZ77+Huffman, Plain LZ77, LZNT1, and their respective decompression algorithms. This algorithm efficiently compresses data that contains repeated byte sequences. It is not designed to compress image, audio, or video data. Between the trade-offs of compressed size and CPU cost, it heavily emphasizes low CPU cost.</p> <p><a href="#">Click here to view this version of the [MS-XCA] PDF.</a></p>
<a href="#">[MS-XCEP]: X.509 Certificate Enrollment Policy Protocol</a>	<p>Specifies the X.509 Certificate Enrollment Policy Protocol. This protocol defines the interactions between a requesting client and a responding server for the exchange of a certificate enrollment policy, which is the collection of certificate templates and certificate issuers available to the requestor for X.509 certificate enrollment.</p> <p><a href="#">Click here to view this version of the [MS-XCEP] PDF.</a></p>
<a href="#">[MS-XOPP]: XML-binary Optimized Packaging (XOP) Profile</a>	<p>Specifies the XML-binary Optimized Packaging (XOP) Profile, which provides extensions that enable more efficient implementations of [XML-XOP] to be built by requiring certain ordering of the MIME parts in the XOP package</p> <p><a href="#">Click here to view this version of the [MS-XOPP] PDF.</a></p>

# [MS-SRVS]: Server Service Remote Protocol

Article 10/04/2021

Specifies the Server Service Remote Protocol, which remotely enables file and printer sharing and named pipe access to the server through the Server Message Block Protocol.

This page and associated content may be updated frequently. We recommend you subscribe to the [RSS feed ↗](#) to receive update notifications.

## Published Version

Date	Protocol Revision	Revision Class	Downloads
10/6/2021	38.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>

[Click here to download a zip file of all PDF files for Windows Protocols. ↗](#)

## Previous Versions

Date	Protocol Revision	Revision Class	Downloads
6/25/2021	37.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>
4/7/2021	36.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>
9/12/2018	35.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>
9/15/2017	34.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>
6/1/2017	33.0	None	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>
7/14/2016	33.0	None	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>   <a href="#">Diff ↗</a>
10/16/2015	33.0	None	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>
6/30/2015	33.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>
5/15/2014	32.0	None	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>
2/13/2014	32.0	None	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>
11/14/2013	32.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>

Date	Protocol Revision	Revision Class	Downloads
8/8/2013	31.0	Major	<a href="#">PDF ↗</a>   <a href="#">DOCX ↗</a>
1/31/2013	30.0	None	
10/25/2012	30.0	Major	
7/12/2012	29.0	None	
3/30/2012	29.0	Major	
12/16/2011	28.0	Major	
9/23/2011	27.0	Major	
6/17/2011	26.1	Minor	
5/6/2011	26.0	Major	
3/25/2011	25.0	Major	
2/11/2011	24.0	Major	
1/7/2011	23.0	Major	
11/19/2010	22.0	Major	
10/8/2010	21.0	Major	
8/27/2010	20.0	Major	
7/16/2010	19.0	Major	
6/4/2010	18.0	Major	
4/23/2010	17.0	Major	
3/12/2010	16.0	Major	
1/29/2010	15.0	Major	
12/18/2009	14.0	Major	
11/6/2009	13.0	Major	
9/25/2009	12.2	Minor	
8/14/2009	12.1.1	Editorial	
7/2/2009	12.1	Minor	
5/22/2009	12.0	Major	

Date	Protocol Revision	Revision Class	Downloads
4/10/2009	11.0	Major	
2/27/2009	10.0	Major	
1/16/2009	9.1	Minor	
12/5/2008	9.0	Major	
10/24/2008	8.1	Minor	
8/29/2008	8.0	Major	
7/25/2008	7.1	Minor	
6/20/2008	7.0	Major	
5/16/2008	6.0	Major	
3/14/2008	5.0	Major	
1/25/2008	4.2.2	Editorial	
11/30/2007	4.2.1	Editorial	
10/23/2007	4.2	Minor	
9/28/2007	4.1	Minor	
8/10/2007	4.0	Major	
7/20/2007	3.0	Major	
7/3/2007	2.0	Major	
6/1/2007	1.3.1	Editorial	
5/11/2007	1.3	Minor	
4/3/2007	1.2	Minor	
3/2/2007	1.1	Minor	
1/19/2007	1.0	Major	
10/22/2006	0.01	New	

## Preview Versions

From time to time, Microsoft may publish a preview, or pre-release, version of an Open Specifications technical document for community review and feedback. To submit feedback for a preview version of a technical document, please follow any instructions specified for that document. If no instructions are indicated for the document, please provide feedback by using the [Open Specification Forums](#).

The preview period for a technical document varies. Additionally, not every technical document will be published for preview.

A preview version of this document may be available on the [Windows Protocols - Preview Documents](#) page. After the preview period, the most current version of the document is available on this page.

## Development Resources

Find resources for creating interoperable solutions for Microsoft software, services, hardware, and non-Microsoft products:

[Plugfests and Events](#), [Test Tools](#), [Development Support](#), and [Open Specifications Dev Center](#).

## Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

# 1 Introduction

Article02/14/2019

This document specifies the Server Service Remote Protocol. The Server Service Remote Protocol is a [remote procedure call \(RPC\)](#)–based protocol that is used for remotely enabling file and printer sharing and [named pipe](#) access to the [server](#) through the Server Message Block (SMB) Protocol, as specified in [\[MS-SMB\]](#). The protocol is also used for remote administration of servers that are running Windows.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

# 1.1 Glossary

Article 10/04/2021

This document uses the following terms:

**client:** A computer on which the remote procedure call (RPC) client is executing.

**connection:** Firewall rules are specified to apply to connections. Every packet is associated with a connection based on TCP, UDP, or IP endpoint parameters; see [\[IANAPORT\]](#).

**connection blocks:** A pre-allocated chunk of memory that is used to store a single connection request.

**Distributed File System (DFS):** A file system that logically groups physical shared folders located on different servers by transparently connecting them to one or more hierarchical namespaces. [DFS](#) also provides fault-tolerance and load-sharing capabilities.

**Distributed File System (DFS) link:** A component in a DFS path that lies below the [DFS root](#) and maps to one or more DFS link targets. Also interchangeably used to refer to a DFS path that contains the [DFS link](#).

**Distributed File System (DFS) root:** The starting point of the DFS namespace. The root is often used to refer to the namespace as a whole. A [DFS root](#) maps to one or more root targets, each of which corresponds to a share on a separate server. A [DFS root](#) has one of the following formats "`\<ServerName>\<RootName>`" or "`\<DomainName>\<RootName>`". Where `<ServerName>` is the name of the root target server hosting the DFS namespace; `<DomainName>` is the name of the domain that hosts the [DFS root](#); and `<RootName>` is the name of the root of a domain-based [DFS](#). The [DFS root](#) must reside on an NTFS volume.

**Domain Name System (DNS):** A hierarchical, distributed database that contains mappings of domain names to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

**endpoint:** A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the [RPC](#) protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence ncacn\_ip\_tcp), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence ncacn\_np), an endpoint might be the name of a [named pipe](#). For more information, see [\[C706\]](#).

**globally unique identifier (GUID)**: A term used interchangeably with [universally unique identifier \(UUID\)](#) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [\[RFC4122\]](#) or [\[C706\]](#) must be used for generating the [GUID](#). See also [universally unique identifier \(UUID\)](#).

**Interface Definition Language (IDL)**: The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [\[C706\]](#) section 4.

**Internet host name**: The name of a host as defined in [\[RFC1123\]](#) section 2.1, with the extensions described in [\[MS-HNDS\]](#).

**mailslot**: A mechanism for one-way interprocess communications (IPC). For more information, see [\[MSLOT\]](#) and [\[MS-MAIL\]](#).

**Microsoft Interface Definition Language (MIDL)**: The Microsoft implementation and extension of the OSF-DCE [Interface Definition Language \(IDL\)](#). [MIDL](#) can also mean the [Interface Definition Language \(IDL\)](#) compiler provided by Microsoft. For more information, see [\[MS-RPCE\]](#).

**named pipe**: A named, one-way, or duplex pipe for communication between a pipe server and one or more pipe clients.

**NetBIOS host name**: The NetBIOS name of a host (as described in [\[RFC1001\]](#) section 14 and [\[RFC1002\]](#) section 4), with the extensions described in [\[MS-NBTE\]](#).

**Quality of Service (QoS)**: A set of technologies that do network traffic manipulation, such as packet marking and reshaping.

**remote procedure call (RPC)**: A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). For more information, see [\[C706\]](#).

**scoped share**: A [share](#) that is only available to a [client](#) if accessed through a specific DNS or NetBIOS name. [Scoped shares](#) can make a single [server](#) appear to be multiple, distinct [servers](#) by providing access to a different set of [shares](#) based on the name the [client](#) uses to access the [server](#).

**server**: A computer on which the [remote procedure call \(RPC\)](#) server is executing.

**Server Message Block (SMB):** A protocol that is used to request file and print services from server systems over a network. The SMB protocol extends the CIFS protocol with additional security, file, and disk management support. For more information, see [\[CIFS\]](#) and [\[MS-SMB\]](#).

**share:** A resource offered by a Common Internet File System (CIFS) server for access by CIFS clients over the network. A **share** typically represents a directory tree and its included files (referred to commonly as a "disk share" or "file share") or a printer (a "print share"). If the information about the **share** is saved in persistent store (for example, Windows registry) and reloaded when a file server is restarted, then the **share** is referred to as a "sticky share". Some **share** names are reserved for specific functions and are referred to as special **shares**: IPC\$, reserved for interprocess communication, ADMIN\$, reserved for remote administration, and A\$, B\$, C\$ (and other local disk names followed by a dollar sign), assigned to local disk devices.

**site:** A group of related webpages that is hosted by a server on the World Wide Web or an intranet. Each website has its own entry points, metadata, administration settings, and workflows. Also referred to as web site.

**standalone DFS implementation:** A Distributed File System (DFS) namespace whose configuration information is stored locally in the registry of the root **server**.

**sticky share:** A **share** that is available after a machine restarts.

**universally unique identifier (UUID):** A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and **RPC** objects. UUIDs are highly likely to be unique. UUIDs are also known as **globally unique identifiers (GUIDs)** and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [\[RFC4122\]](#) or [\[C706\]](#) must be used for generating the UUID.

**work item:** A buffer that receives a user request, which is held by the Server Message Block (SMB) **server** while it is being processed.

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

## 1.2 References

Article02/14/2019

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](#).

## 1.2.1 Normative References

Article 10/04/2021

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](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997,  
[https://publications.opengroup.org/c706 ↗](https://publications.opengroup.org/c706)

**Note** Registration is required to download the document.

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

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

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

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

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

[MS-EERR] Microsoft Corporation, "[ExtendedError Remote Data Structure](#)".

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](#)".

[MS-NRPC] Microsoft Corporation, "[Netlogon Remote Protocol](#)".

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

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

[MS-SMB] Microsoft Corporation, "[Server Message Block \(SMB\) Protocol](#)".

[RFC1001] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", RFC 1001, March 1987,  
[http://www.ietf.org/rfc/rfc1001.txt ↗](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.rfc-editor.org/rfc/rfc1002.txt ↗](http://www.rfc-editor.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>

[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>

## 1.2.2 Informative References

Article02/14/2019

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

[MSDN-CoCreateGuid] Microsoft Corporation, "CoCreateGuid function",  
[http://msdn.microsoft.com/en-us/library/ms688568.aspx ↗](http://msdn.microsoft.com/en-us/library/ms688568.aspx)

[NWLINK] Microsoft Corporation, "Description of Microsoft NWLINK IPX/SPX-Compatible Transport", October 2006, [http://support.microsoft.com/?kbid=203051 ↗](http://support.microsoft.com/?kbid=203051)

[OFFLINE] Microsoft Corporation, "Offline Files", January 2005,  
[http://technet2.microsoft.com/WindowsServer/en/Library/830323a2-23ca-4875-af3c-06671d68ca9a1033.mspx ↗](http://technet2.microsoft.com/WindowsServer/en/Library/830323a2-23ca-4875-af3c-06671d68ca9a1033.mspx)

[PIPE] Microsoft Corporation, "Named Pipes", [http://msdn.microsoft.com/en-us/library/aa365590.aspx ↗](http://msdn.microsoft.com/en-us/library/aa365590.aspx)

# 1.3 Overview

Article 10/30/2020

The Server Service Remote Protocol is designed for remotely querying and configuring a [Server Message Block \(SMB\) server](#) on a remote computer. By using this protocol, a [client](#) can query and configure information on the server such as active [connections](#), sessions, shares, files, and transport protocols. Clients can also query and configure the server itself, for instance by setting the server's type, changing the services that are running on the server, or getting a list of all servers of a specific type in a domain.

A server can be configured to present different resources based on the name the client connects with, allowing it to appear as multiple, distinct servers. This is achieved by scoping a share to a specific name, and hosting all of the names on the same server.

The server can also configure one or more aliases, identifying that multiple distinct names present the same resources. For example, the administrator could choose to expose the same shares for the name "server" and "server.example.com" by creating an alias indicating that "server.example.com" is the same as "server". The SMB client will connect using the name provided by the calling applications, and is not aware whether the name is the server's default machine name, an additionally configured name, or an alias. For more information, see the example in section [4.3](#).

This is an [RPC](#)-based protocol. The server does not maintain client state information. No sequence of method calls is imposed on this protocol, with the exception of net share deletion, which requires a two-phase commit, net file get information, and net file close.

# 1.4 Relationship to Other Protocols

Article02/14/2019

This protocol depends on [RPC](#) and [SMB](#) for its transport. This protocol uses RPC over [named pipes](#), as specified in section [2.1](#). Named pipes use the SMB protocols, as specified in [\[MS-CIFS\]](#), [\[MS-SMB\]](#), and [\[MS-SMB2\]](#).

This protocol calls the Common Internet File System (CIFS) Protocol, the Server Message Block (SMB) Protocol, or the SMB Version 2 Protocol for file server management.

CIFS, SMB, and SMB Version 2 call the Server Service Remote Protocol for synchronizing the information on shares, sessions, treeconnects, file opens, and server configuration. The synchronization mechanism is dependent upon CIFS, SMB, SMB2 servers, and the server service starting up and terminating together, in order to share and maintain a consistent view of the common data among all protocols at all times.

This protocol calls the DFS Namespace Management Protocol, as specified in [\[MS-DFSNM\]](#), to identify a [DFS](#) share.

# 1.5 Prerequisites/Preconditions

Article02/14/2019

The Server Service Remote Protocol is an [RPC](#) interface and, as a result, has the prerequisites that are described in [\[MS-RPCE\]](#) section [1.5](#) as being common to RPC interfaces.

It is assumed that a Server Service Remote Protocol [client](#) has obtained the name of a remote machine that supports the Server Service Remote Protocol before this protocol is invoked. This specification does not describe how a client invokes this protocol.

# 1.6 Applicability Statement

Article04/06/2021

The Server Service Remote Protocol is applicable to environments that require management and monitoring of a file [server](#). In particular, this protocol provides for the creation, deletion, and management of file shares on the server and the monitoring and administering of users who access that file server. Therefore, this protocol is applicable to environments that require those features.

The Server Service Remote Protocol is used for the management of file servers that use the SMB Protocol, as specified in [\[MS-SMB\]](#).

# 1.7 Versioning and Capability Negotiation

Article02/14/2019

None.

# 1.8 Vendor-Extensible Fields

Article02/14/2019

This protocol does not define any vendor-extensible fields.

This protocol uses Win32 error codes. These values are taken from the Windows error number space defined in [\[MS-EERR\]](#). Vendors SHOULD reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision in the future.

<1>

# 1.9 Standards Assignments

Article 10/30/2020

Parameter	Value	Reference
RPC Interface UUID	4b324fc8-1670-01d3-1278-5a47bf6ee188	Section 2.1
Pipe Name	\PIPE\svrsvc	Section 2.1

# 2 Messages

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [2.1 Transport](#)
- [2.2 Common Data Types](#)

# 2.1 Transport

Article 10/30/2020

The [RPC](#) methods that the Server Service Remote Protocol exposes are available on one endpoint:

- [srsvc named pipe](#) (RPC protseqs ncacn\_np), as specified in [\[MS-RPCE\]](#) section [2.1.1.2](#).

The Server Service Remote Protocol endpoint is available only over named pipes. For more details about named pipes, see [\[PIPE\]](#).

This protocol MUST use the [UUID](#) as specified in section [1.9](#). The RPC version number is 3.0.

This protocol allows any user to establish a [connection](#) to the [RPC server](#). The protocol uses the underlying RPC protocol to retrieve the identity of the caller that made the method call, as specified in [\[MS-RPCE\]](#) section [3.3.3.4.3](#). The server SHOULD use this identity to perform method-specific access checks as specified in section [3.1.4.<2>](#)

## 2.2 Common Data Types

Article 10/30/2020

In addition to [RPC base types defined in \[C706\]](#) and [\[MS-RPCE\]](#), the data types that follow are defined in the [Microsoft Interface Definition Language \(MIDL\)](#) specification for this RPC interface.

This protocol uses the following types, as specified in [\[MS-DTYP\]](#).

Type	Reference
<a href="#">DWORD</a>	[MS-DTYP] section 2.2.9
<a href="#">GUID</a>	[MS-DTYP] section 2.3.4
<a href="#">NET_API_STATUS</a>	[MS-DTYP] section 2.2.37
<a href="#">SECURITY_INFORMATION</a>	[MS-DTYP] section 2.4.7
<a href="#">WCHAR</a>	[MS-DTYP] section 2.2.60

## 2.2.1 Simple Data Types

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [2.2.1.1 SRVSVC\\_HANDLE](#)
- [2.2.1.2 SHARE\\_DEL\\_HANDLE](#)
- [2.2.1.3 PSHARE\\_DEL\\_HANDLE](#)

## 2.2.1.1 SRVSVC\_HANDLE

Article02/14/2019

SRVSVC\_HANDLE: A pointer to a null-terminated Unicode UTF-16 string that specifies the [Internet host name](#) or [NetBIOS host name](#) of the remote server on which the method is to execute that is pre-pended with "\\\" (two literal backslash characters).

This type is declared as follows:

```
typedef [handle, string] wchar_t* SRVSVC_HANDLE;
```

## 2.2.1.2 SHARE\_DEL\_HANDLE

Article02/14/2019

SHARE\_DEL\_HANDLE: An [RPC](#) context handle, as specified in [\[C706\]](#) section 6, returned by the [NetrShareDelStart](#) method, to be provided as a parameter to the [NetrShareDelCommit](#) method.

This type is declared as follows:

```
typedef [context_handle] void* SHARE_DEL_HANDLE;
```

## 2.2.1.3 PSHARE\_DEL\_HANDLE

Article02/14/2019

**PSHARE\_DEL\_HANDLE:** A pointer to a [SHARE\\_DEL\\_HANDLE \(section 2.2.1.2\)](#) datatype.

This type is declared as follows:

```
typedef SHARE_DEL_HANDLE* PSHARE_DEL_HANDLE;
```

## 2.2.2 Constants

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [2.2.2.1 Sessionclient Types](#)
- [2.2.2.2 MAX\\_PREFERRED\\_LENGTH](#)
- [2.2.2.3 Session User Flags](#)
- [2.2.2.4 Share Types](#)
- [2.2.2.5 Client-Side Caching \(CSC\) States](#)
- [2.2.2.6 Platform IDs](#)
- [2.2.2.7 Software Type Flags](#)
- [2.2.2.8 Name Types](#)
- [2.2.2.9 Path Types](#)
- [2.2.2.10 Common Error Codes](#)
- [2.2.2.11 SHARE\\_INFO Parameter Error Codes](#)
- [2.2.2.12 SERVER\\_INFO Parameter Error Codes](#)
- [2.2.2.13 DFS Entry Flags](#)

## 2.2.2.1 Sessionclient Types

Article02/14/2019

Sessionclient is a Unicode UTF-16 string value that is used to specify the type of [client](#) that established the session.<3>

The client generates an implementation-defined string that describes the client operating system version. The server SHOULD NOT enforce any limits on the Sessionclient string length.<4>

## 2.2.2.2 MAX\_PREFERRED\_LENGTH

Article10/30/2020

The following table describes the MAX\_PREFERRED\_LENGTH constant.

Constant/value	Description
MAX_PREFERRED_LENGTH -1	A constant of type DWORD that is set to –1. This value is valid as an input parameter to any method in section <a href="#">3.1.4</a> that takes a <i>PreferredMaximumLength</i> parameter. When specified as an input parameter, this value indicates that the method MUST allocate as much space as the data requires.

## 2.2.2.3 Session User Flags

Article10/30/2020

The following flags specify information that is related to how a user established a session.

Constant/value	Description
SESS_GUEST 0x00000001	The user specified by the sesi*_username member established the session by using a guest account.
SESS_NOENCRYPTION 0x00000002	The user specified by the sesi*_username member established the session without using password encryption.

## 2.2.2.4 Share Types

Article 10/30/2020

The following values are used to specify the type of a shared resource.

Constant/value	Description
STYPE_DISKTREE 0x00000000	Disk drive
STYPE_PRINTQ 0x00000001	Print queue
STYPE_DEVICE 0x00000002	Communication device
STYPE_IPC 0x00000003	Interprocess communication (IPC)
STYPE_CLUSTER_FS 0x02000000	A cluster share
STYPE_CLUSTER_SOFS 0x04000000	A Scale-Out cluster share
STYPE_CLUSTER_DFS 0x08000000	A DFS share in a cluster

The following table of values can be OR'd with the values in the preceding table to further specify the characteristics of a shared resource. It is possible to use both values in this OR operation.

Constant/value	Description
STYPE_SPECIAL 0x80000000	Special share reserved for interprocess communication (IPC\$) or remote administration of the <a href="#">server</a> (ADMIN\$). Can also refer to administrative shares such as C\$, D\$, E\$, and so forth.
STYPE_TEMPORARY 0x40000000	A temporary share that is not persisted for creation each time the file server initializes.

## 2.2.2.5 Client-Side Caching (CSC) States

Article10/04/2021

The following values are used to specify states that provide hints to [clients](#) about whether to cache files by using client-side caching with the [SMB Protocol](#), as specified in [\[MS-SMB\]](#).

Constant/value	Description
CSC_CACHE_MANUAL_REINT 0x00	The client MUST allow only manual caching for the files open from this share.
CSC_CACHE_AUTO_REINT 0x10	The client MAY cache every file that it opens from this share.
CSC_CACHE_VDO 0x20	The client MAY cache every file that it opens from this share. Also, the client MAY satisfy the file requests from its local cache.
CSC_CACHE_NONE 0x30	The client MUST NOT cache any files from this share.

## 2.2.2.6 Platform IDs

Article 10/30/2020

The following values are returned by the [server](#) to indicate its platform version.<5><6>

Constant/value	Description
PLATFORM_ID_DOS 300	Specified by a server running DOS.
PLATFORM_ID_OS2 400	Specified by a server running OS2.
PLATFORM_ID_NT 500	Specified by a server running Windows NT or a newer Windows operating system version.
PLATFORM_ID_OSF 600	Specified by a server running OSF/1.
PLATFORM_ID_VMS 700	Specified by a server running VMS.

## 2.2.2.7 Software Type Flags

Article 04/06/2021

The SV\_TYPE flags indicate the services that are available on the [server](#).

Constant/value	Description
SV_TYPE_WORKSTATION 0x00000001	A server running the WorkStation Service.
SV_TYPE_SERVER 0x00000002	A server running the Server Service.
SV_TYPE_SQLSERVER 0x00000004	A server running SQL Server.
SV_TYPE_DOMAIN_CTRL 0x00000008	A primary domain controller.
SV_TYPE_DOMAIN_BAKCTRL 0x00000010	A backup domain controller.
SV_TYPE_TIME_SOURCE 0x00000020	A server is available as a time source for network time synchronization.
SV_TYPE_AFP 0x00000040	An Apple File Protocol server.
SV_TYPE_NOVELL 0x00000080	A Novell server.
SV_TYPE_DOMAIN_MEMBER 0x00000100	A LAN Manager 2.x domain member.
SV_TYPE_PRINTQ_SERVER 0x00000200	A server sharing print queue.
SV_TYPE_DIALIN_SERVER 0x00000400	A server running a dial-in service.

Constant/value	Description
SV_TYPE_XENIX_SERVER 0x00000800	A Xenix server.
SV_TYPE_NT 0x00001000	Windows Server 2003 operating system, Windows XP operating system, Windows 2000 operating system, or Windows NT operating system.
SV_TYPE_WFW 0x00002000	A server running Windows for Workgroups.
SV_TYPE_SERVER_MFPN 0x00004000	Microsoft File and Print for NetWare.
SV_TYPE_SERVER_NT 0x00008000	Windows Server 2003, Windows 2000 Server operating system, or a server that is not a domain controller.
SV_TYPE_POTENTIAL_BROWSER 0x00010000	A server that can run the browser service.
SV_TYPE_BACKUP_BROWSER 0x00020000	A server running a browser service as backup.
SV_TYPE_MASTER_BROWSER 0x00040000	A server running the master browser service.
SV_TYPE_DOMAIN_MASTER 0x00080000	A server running the domain master browser.
SV_TYPE_WINDOWS 0x00400000	Windows Millennium Edition operating system, Windows 98 operating system, or Windows 95 operating system.
SV_TYPE_DFS 0x00800000	A server running the DFS service.
SV_TYPE_CLUSTER_NT 0x01000000	Server clusters available in the domain.
SV_TYPE_TERMINALSERVER 0x02000000	Terminal Server.

Constant/value	Description
SV_TYPE_CLUSTER_VS_NT 0x04000000	Cluster virtual servers available in the domain.
SV_TYPE_DCE 0x10000000	A server running IBM DSS (Directory and Security Services) or equivalent.
SV_TYPE_ALTERNATE_XPORT 0x20000000	Return list for alternate transport.
SV_TYPE_LOCAL_LIST_ONLY 0x40000000	Servers maintained by the browser.
SV_TYPE_DOMAIN_ENUM 0x80000000	Primary domain.
SV_TYPE_ALL 0xFFFFFFFF	All servers.

## 2.2.2.8 Name Types

Article 10/30/2020

The following values specify types of names that are used with the [NetprNameValidate](#), [NetprNameCanonicalize](#), and [NetprNameCompare](#) methods.

Constant/value	Description
NAMETYPE_USER 1	User name
NAMETYPE_PASSWORD 2	User password
NAMETYPE_GROUP 3	Group name
NAMETYPE_COMPUTER 4	Computer name
NAMETYPE_EVENT 5	Event name
NAMETYPE_DOMAIN 6	NetBIOS name of a domain
NAMETYPE_SERVICE 7	Service name
NAMETYPE_NET 8	Net name
NAMETYPE_SHARE 9	Share name
NAMETYPE_MESSAGE 10	Message name
NAMETYPE_MESSAGEDEST 11	Message destination

Constant/value	Description
NAMETYPE_SHAREPASSWORD 12	Share password
NAMETYPE_WORKGROUP 13	Workgroup name

More information for each NameType is listed following.

The set of default invalid characters includes "`\[]:<>+=;?` as well as the control characters in the range from `0x01` through `0x1F`, inclusive.

Constant	Min/max length	Invalid characters	Restricted to dots and spaces?	Other requirements
NAMETYPE_USER	1/256	Default	No	
NAMETYPE_PASSWORD	0/256	0x00	Yes	
NAMETYPE_GROUP	1/256		Default	No
NAMETYPE_COMPUTER	1/260	Default and *	no	No leading or trailing blanks.
NAMETYPE_EVENT	1/16	Default	No	
NAMETYPE_DOMAIN	1/15	Default, *, 0x20	No	
NAMETYPE_SERVICE	1/80	Default	No	
NAMETYPE_NET	1/260	Default	No	
NAMETYPE_SHARE	1/80	Default	No	
NAMETYPE_MESSAGE	1/15	Default	No	
NAMETYPE_MESSAGEDEST	1/260	Default	No	"*" is allowed only as the last character, and names of the maximum length must contain a trailing "*".
NAMETYPE_SHAREPASSWORD	0/8	0x00	Yes	
NAMETYPE_WORKGROUP	1/15	Default	No	

## 2.2.2.9 Path Types

Article 10/30/2020

The following values specify types of paths used with the [NetprPathType](#), [NetprPathCanonicalize](#), and [NetprPathCompare](#) methods.

Constant/value	Description
ITYPE_UNC_COMPNAME 4144	UNC ComputerName
ITYPE_UNC_WC 4145	UNC Wild Card ComputerName
ITYPE_UNC 4096	UNC Path; MUST NOT end with \
ITYPE_UNC_WC_PATH 4097	UNC Path and WC (? or *)
ITYPE_UNC_SYS_SEM 6400	UNC Semaphore
ITYPE_UNC_SYS_SHMEM 6656	UNC Shared Memory
ITYPE_UNC_SYS_MSLOT 6144	UNC <a href="#">Mailslot</a>
ITYPE_UNC_SYS_PIPE 6912	UNC Pipe
ITYPE_UNC_SYS_QUEUE 7680	UNC Queue
ITYPE_PATH_ABSND 8194	Absolute non dot path
ITYPE_PATH_ABSD 8198	Path beginning with \\. or <drive>:\

Constant/value	Description
ITYPE_PATH_RELND 8192	Relative path non dot
ITYPE_PATH_RELD 8196	Relative path beginning with \\.
ITYPE_PATH_ABSND_WC 8195	ITYPE_PATH_ABSND and WC
ITYPE_PATH_ABSD_WC 8199	ITYPE_PATH_ABSD and WC(?) or *
ITYPE_PATH_RELND_WC 8193	ITYPE_PATH_RELND and WC
ITYPE_PATH_RELD_WC 8197	ITYPE_PATH_RELD and WC
ITYPE_PATH_SYS_SEM 10498	Local System Semaphore\path
ITYPE_PATH_SYS_SHMEM 10754	Local System Shared Memory\path
ITYPE_PATH_SYS_MSLOT 10242	Local System Mailslot\path
ITYPE_PATH_SYS_PIPE 11010	Local System Pipe\path
ITYPE_PATH_SYS_COMM 11266	Local System COMM\path
ITYPE_PATH_SYS_PRINT 11522	Local System PRINT\path
ITYPE_PATH_SYS_QUEUE 11778	Local System QUEUE\path

Constant/value	Description
ITYPE_PATH_SYS_SEM_M 43266	Local System Semaphore
ITYPE_PATH_SYS_SHMEM_M 43522	Local System Shared Memory
ITYPE_PATH_SYS_MSLOT_M 43010	Local System Mailslot
ITYPE_PATH_SYS_PIPE_M 43778	Local System Pipe
ITYPE_PATH_SYS_COMM_M 44034	Local System COMM
ITYPE_PATH_SYS_PRINT_M 44290	Local System PRINT
ITYPE_PATH_SYS_QUEUE_M 44546	Local System QUEUE
ITYPE_DEVICE_DISK 16384	<drive>:
ITYPE_DEVICE_LPT 16400	LPT[1-9][:] or \DEV\LPT[1-9]
ITYPE_DEVICE_COM 16416	COM[1-9][:] or \DEV\COM[1-9]
ITYPE_DEVICE_CON 16448	CON port
ITYPE_DEVICE_NUL 16464	NULL port

## 2.2.2.10 Common Error Codes

Article 10/04/2021

The following error codes are referenced in this specification.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	The user does not have access to the requested information.
0x0000007C ERROR_INVALID_LEVEL	The value that is specified for the level parameter is invalid.
0x00000057 ERROR_INVALID_PARAMETER	One or more of the specified parameters is invalid.
0x000000EA ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
0x00000000 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the file specified.
0x00000034 ERROR_DUP_NAME	A duplicate name exists on the network.
0x000004BC ERROR_INVALID_DOMAINNAME	The format of the specified NetBIOS name of a domain is invalid.
0x00000032 ERROR_NOT_SUPPORTED	The server does not support branch cache.
0x00000424 ERROR_SERVICE_DOES_NOT_EXIST	The branch cache component does not exist as an installed service.
0x0000084B NERR_BufTooSmall	The client request succeeded. More entries are available. The buffer size that is specified by <i>PreferredMaximumLength</i> was too small to fit even a single entry.

Return value/code	Description
0x00000908  NERR_ClientNameNotFound	A session does not exist with the computer name.
0x0000092F  NERR_InvalidComputer	The computer name is not valid.
0x000008AD  NERR_UserNotFound	The user name could not be found.
0x00000846  NERR_DuplicateShare	The share name is already in use on this server.
0x00000845  NERR_RedirectedPath	The operation is not valid for a redirected resource. The specified device name is assigned to a shared resource.
0x00000844  NERR_UnknownDevDir	The device or directory does not exist.
0x00000906  NERR_NetNameNotFound	The share name does not exist.
0x00000907  NERR_DeviceNotShared	The device is not shared.
0x00000846  NERR_DuplicateShare	The alias already exists.

## 2.2.2.11 SHARE\_INFO Parameter Error Codes

Article10/30/2020

When an invalid value is specified for a field of the **SHARE\_INFO** structure, one of the following values MUST be used to indicate which field contains an invalid value. In the following table, "\*" is a wildcard character.

Return value/code	Description
1 SHARE_NETNAME_PARMNUM	Indicates that a shi*_netname member caused the error.
3 SHARE_TYPE_PARMNUM	Indicates that a shi*_type member caused the error.
4 SHARE_REMARK_PARMNUM	Indicates that a shi*_remark member caused the error.
5 SHARE_PERMISSIONS_PARMNUM	Indicates that a shi*_permissions member caused the error.
6 SHARE_MAXUSES_PARMNUM	Indicates that a shi*_max_uses member caused the error.
7 SHARE_CURRENTUSES_PARMNUM	Indicates that a shi*_current_uses member caused the error.
8 SHARE_PATH_PARMNUM	Indicates that a shi*_path member caused the error.
9 SHARE_PASSWD_PARMNUM	Indicates that a shi*_passwd member caused the error.
501 SHARE_FILE_SD_PARMNUM	Indicates that a shi*_security_descriptor member caused the error.

## 2.2.2.12 SERVER\_INFO Parameter Error Codes

Article10/04/2021

When an invalid value is specified for a field of the [SERVER\\_INFO](#) structure, one of the following values MUST be used to indicate which field contains an invalid value. In the following table, "\*" is a wildcard character.

Return value/code	Description
101 SV_PLATFORM_ID_PARMNUM	Indicates that a sv*_platform_id member caused the error.
102 SV_NAME_PARMNUM	Indicates that a sv*_name member member caused the error.
103 SV_VERSION_MAJOR_PARMNUM	Indicates that a sv*_version_major member caused the error.
104 SV_VERSION_MINOR_PARMNUM	Indicates that a sv*_version_minor member caused the error.
105 SV_TYPE_PARMNUM	Indicates that a sv*_type member caused the error.
5 SV_COMMENT_PARMNUM	Indicates that a sv*_comment member caused the error.
107 SV_USERS_PARMNUM	Indicates that a sv*_users member caused the error.
10 SV_DISC_PARMNUM	Indicates that a sv*_disc member caused the error.
16 SV_HIDDEN_PARMNUM	Indicates that a sv*_hidden member caused the error.
17 SV_ANNOUNCE_PARMNUM	Indicates that a sv*_announce member caused the error.

Return value/code	Description
18 SV_ANNDELTA_PARMNUM	Indicates that a sv*_anndelta member caused the error.
112 SV_USERPATH_PARMNUM	Indicates that a sv*_userpath member caused the error.
501 SV_SESSOPENS_PARMNUM	Indicates that a sv*_sessopens member caused the error.
502 SV_SESSVCS_PARMNUM	Indicates that a sv*_sessvcs member caused the error.
503 SV_OPENSEARCH_PARMNUM	Indicates that a sv*_opensearch member caused the error.
504 SV_SIZREQBUF_PARMNUM	Indicates that a sv*_sizreqbuf member caused the error.
505 SV_INITWORKITEMS_PARMNUM	Indicates that a sv*_initworkitems member caused the error.
506 SV_MAXWORKITEMS_PARMNUM	Indicates that a sv*_maxworkitems member caused the error.
507 SV_RAWWORKITEMS_PARMNUM	Indicates that a sv*_rawworkitems member caused the error.
508 SV_IRPSTACKSIZE_PARMNUM	Indicates that a sv*_irpstacksize member caused the error.
509 SV_MAXRAWBUFLEN_PARMNUM	Indicates that a sv*_maxrawbuflen member caused the error.
510 SV_SESSUSERS_PARMNUM	Indicates that a sv*_sessusers member caused the error.
511 SV_SESSCONNS_PARMNUM	Indicates that a sv*_sessconns member caused the error.

Return value/code	Description
512 SV_MAXNONPAGEDMEMORYUSAGE_PARMNUM	Indicates that a sv*_maxnonpagedmemoryusage member caused the error.
513 SV_MAXPAGEDMEMORYUSAGE_PARMNUM	Indicates that a sv*_maxpagedmemoryusage member caused the error.
514 SV_ENABLESOFTCOMPAT_PARMNUM	Indicates that a sv*_enablesoftcompat member caused the error.
515 SV_ENABLEFORCEDLOGOFF_PARMNUM	Indicates that a sv*_enableforcedlogoff member caused the error.
516 SV_TIMESOURCE_PARMNUM	Indicates that a sv*_timesource member caused the error.
517 SV_ACCEPTDOWNLEVELAPIS_PARMNUM	Indicates that a sv*_acceptdownlevelapis member caused the error.
518 SV_LMANOUNCE_PARMNUM	Indicates that a sv*_lmannounce member caused the error.
519 SV_DOMAIN_PARMNUM	Indicates that a sv*_domain member caused the error.
520 SV_MAXCOPYREADLEN_PARMNUM	Indicates that a sv*_maxcopyreadlen member caused the error.
521 SV_MAXCOPYWRITELEN_PARMNUM	Indicates that a sv*_maxcopywritelen member caused the error.
522 SV_MINKEEPSEARCH_PARMNUM	Indicates that a sv*_minkeepsearch member caused the error.
523 SV_MAXKEEPSEARCH_PARMNUM	Indicates that a sv*_maxkeepsearch member caused the error.
524 SV_MINKEEPCOMPLSEARCH_PARMNUM	Indicates that a sv*_minkeepcomplsearch member caused the error.

Return value/code	Description
525 SV_MAXKEEPCOMPLSEARCH_PARMNUM	Indicates that a sv*_maxkeepcomplsearch member caused the error.
526 SV_THREADCOUNTADD_PARMNUM	Indicates that a sv*_threadcountadd member caused the error.
527 SV_NUMBLOCKTHREADS_PARMNUM	Indicates that a sv*_numblockthreads member caused the error.
528 SV_SCAVTIMEOUT_PARMNUM	Indicates that a sv*_scavtimeout member caused the error.
529 SV_MINRCVQUEUE_PARMNUM	Indicates that a sv*_minrcvqueue member caused the error.
530 SV_MINFREEWORKITEMS_PARMNUM	Indicates that a sv*_minfreeworkitems member caused the error.
531 SV_XACTMEMSIZE_PARMNUM	Indicates that a sv*_xactmemsize member caused the error.
532 SV_THREADPRIORITY_PARMNUM	Indicates that a sv*_threadpriority member caused the error.
533 SV_MAXMPXCT_PARMNUM	Indicates that a sv*_maxmpxct member caused the error.
534 SV_OPLOCKBREAKWAIT_PARMNUM	Indicates that a sv*_oplockbreakwait member caused the error.
535 SV_OPLOCKBREAKRESPONSEWAIT_PARMNUM	Indicates that a sv*_oplockbreakresponsewait member caused the error.
536 SV_ENABLEOPLOCKS_PARMNUM	Indicates that a sv*_enableoplocks member caused the error.
537 SV_ENABLEOPLOCKFORCECLOSE_PARMNUM	Indicates that a sv*_enableoplockforceclose member caused the error.

Return value/code	Description
538 SV_ENABLEFCBOPENS_PARMNUM	Indicates that a sv*_enablefcopens member caused the error.
539 SV_ENABLERAW_PARMNUM	Indicates that a sv*_enableraw member caused the error.
540 SV_ENABLESHAREDNETDRIVES_PARMNUM	Indicates that a sv*_enablesharednetdrives member caused the error.
541 SV_MINFREECONNECTIONS_PARMNUM	Indicates that a sv*_minfreeconnections member caused the error.
542 SV_MAXFREECONNECTIONS_PARMNUM	Indicates that a sv*_maxfreeconnections member caused the error.
543 SV_INITSESSTABLE_PARMNUM	Indicates that a sv*_initsesstable member caused the error.
544 SV_INITCONNTABLE_PARMNUM	Indicates that a sv*_initconntable member caused the error.
545 SV_INITFILETABLE_PARMNUM	Indicates that a sv*_initfiletable member caused the error.
546 SV_INITSEARCHTABLE_PARMNUM	Indicates that a sv*_initsearchtable member caused the error.
547 SV_ALERTSCHEDULE_PARMNUM	Indicates that a sv*_alertschedule member caused the error.
548 SV_ERRORTHRESHOLD_PARMNUM	Indicates that a sv*_errorthreshold member caused the error.
549 SV_NETWORKERRORTHRESHOLD_PARMNUM	Indicates that a sv*_networkerrorthreshold member caused the error.
550 SV_DISKSPACETHRESHOLD_PARMNUM	Indicates that a sv*_diskspacethreshold member caused the error.

Return value/code	Description
552 SV_MAXLINKDELAY_PARMNUM	Indicates that a sv*_maxlinkdelay member caused the error.
553 SV_MINLINKTHROUGHPUT_PARMNUM	Indicates that a sv*_minlinkthroughput member caused the error.
554 SV_LINKINFOVALIDTIME_PARMNUM	Indicates that a sv*_linkinfovalidtime member caused the error.
555 SV_SCAVQOSINFOUPDATETIME_PARMNUM	Indicates that a sv*_scavqosinfoupdatetime member caused the error.
556 SV_MAXWORKITEMIDLETIME_PARMNUM	Indicates that a sv*_maxworkitemidletime member caused the error.

## 2.2.2.13 DFS Entry Flags

Article10/04/2021

The following flags specify the details about a [DFS](#) entry that an [SMB file server](#) maintains. For more details about DFS entries, see [\[MS-DFSC\]](#).

Constant/value	Description
PKT_ENTRY_TYPE_CAIRO 0x0001	Entry refers to a particular machine.<7>
PKT_ENTRY_TYPE_MACHINE 0x0002	Entry is a machine volume.
PKT_ENTRY_TYPE_NONCAIRO 0x0004	Entry refers to a server running a pre-Windows NT operating system version of Windows.
PKT_ENTRY_TYPE_LEAFONLY 0x0008	Entry is a <a href="#">DFS link</a> .
PKT_ENTRY_TYPE_OUTSIDE_MY_DOM 0x0010	Entry refers to volume in a foreign domain.
PKT_ENTRY_TYPE_INSITE_ONLY 0x0020	Only give Active Directory in-site referrals.
PKT_ENTRY_TYPE_REFERRAL_SVC 0x0080	Entry refers to a <a href="#">DFS root</a> .
PKT_ENTRY_TYPE_PERMANENT 0x0100	Entry cannot be scavenged.
PKT_ENTRY_TYPE_LOCAL 0x0400	Entry refers to local volume.
PKT_ENTRY_TYPE_LOCAL_XPOINT 0x0800	Entry refers to an exit point.
PKT_ENTRY_TYPE_MACH_SHARE 0x1000	Entry refers to a private machine share.

Constant/value	Description
PKT_ENTRY_TYPE_OFFLINE	Entry refers to a volume that is offline.
0x2000	

## 2.2.3 Unions

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [2.2.3.1 CONNECT\\_ENUM\\_UNION](#)
- [2.2.3.2 FILE\\_ENUM\\_UNION](#)
- [2.2.3.3 FILE\\_INFO](#)
- [2.2.3.4 SESSION\\_ENUM\\_UNION](#)
- [2.2.3.5 SHARE\\_ENUM\\_UNION](#)
- [2.2.3.6 SHARE\\_INFO](#)
- [2.2.3.7 SERVER\\_INFO](#)
- [2.2.3.8 SERVER\\_XPORT\\_ENUM\\_UNION](#)
- [2.2.3.9 TRANSPORT\\_INFO](#)
- [2.2.3.10 SERVER\\_ALIAS\\_INFO](#)

## 2.2.3.1 CONNECT\_ENUM\_UNION

Article02/14/2019

The CONNECT\_ENUM\_UNION union contains information about a [connection](#). It is used in the definition of the [CONNECTION\\_ENUM\\_STRUCT](#) structure.

```
typedef
[switch_type(DWORD)]
union _CONNECT_ENUM_UNION {
    [case(0)]
    CONNECT_INFO_0_CONTAINER* Level0;
    [case(1)]
    CONNECT_INFO_1_CONTAINER* Level1;
} CONNECT_ENUM_UNION;
```

**Level0:** A pointer to a structure containing information about a connection, as specified in section [2.2.4.3](#).

**Level1:** A pointer to a structure containing information about a connection, as specified in section [2.2.4.4](#).

## 2.2.3.2 FILE\_ENUM\_UNION

Article02/14/2019

The FILE\_ENUM\_UNION union contains information about files, devices, and pipes. It is used in the definition of the [FILE\\_ENUM\\_STRUCT](#) structure.

```
typedef
[switch_type(DWORD)]
union _FILE_ENUM_UNION {
    [case(2)]
        FILE_INFO_2_CONTAINER* Level2;
    [case(3)]
        FILE_INFO_3_CONTAINER* Level3;
} FILE_ENUM_UNION;
```

**Level2:** A pointer to a structure containing information about a file, device or pipe, as specified in section [2.2.4.8](#).

**Level3:** A pointer to a structure containing information about a file, device or pipe, as specified in section [2.2.4.9](#).

## 2.2.3.3 FILE\_INFO

Article04/06/2021

The FILE\_INFO union contains information about a file, device, or pipe. This union is used by the [NetrFileGetInfo](#) method.

```
typedef
[switch_type(unsigned long)]
union _FILE_INFO {
    [case(2)]
    LPFILE_INFO_2 FileInfo2;
    [case(3)]
    LPFILE_INFO_3 FileInfo3;
} FILE_INFO,
*PFILE_INFO,
*LPFILE_INFO;
```

**FileInfo2:** A pointer to a structure that contains information about a file, device, or pipe. For more details, see [FILE\\_INFO\\_2 \(section 2.2.4.6\)](#).

**FileInfo3:** A pointer to a structure that contains information about a file, device, or pipe. For more details, see [FILE\\_INFO\\_3 \(section 2.2.4.7\)](#).

## 2.2.3.4 SESSION\_ENUM\_UNION

Article02/14/2019

The SESSION\_ENUM\_UNION union contains information about sessions. It is used in the definition of the [SESSION\\_ENUM\\_STRUCT](#) structure.

```
typedef
[switch_type(DWORD)]
union _SESSION_ENUM_UNION {
    [case(0)]
        SESSION_INFO_0_CONTAINER* Level0;
    [case(1)]
        SESSION_INFO_1_CONTAINER* Level1;
    [case(2)]
        SESSION_INFO_2_CONTAINER* Level2;
    [case(10)]
        SESSION_INFO_10_CONTAINER* Level10;
    [case(502)]
        SESSION_INFO_502_CONTAINER* Level502;
} SESSION_ENUM_UNION;
```

**Level0:** A pointer to a structure that contains information about sessions, as specified in section [2.2.4.16](#).

**Level1:** A pointer to a structure that contains information about sessions, as specified in section [2.2.4.17](#).

**Level2:** A pointer to a structure that contains information about sessions, as specified in section [2.2.4.18](#).

**Level10:** A pointer to a structure that contains information about sessions, as specified in section [2.2.4.19](#).

**Level502:** A pointer to a structure that contains information about sessions, as specified in section [2.2.4.20](#).

## 2.2.3.5 SHARE\_ENUM\_UNION

Article 02/14/2019

The SHARE\_ENUM\_UNION union contains information about shares. It is used in the definition of the [SHARE\\_ENUM\\_STRUCT](#) structure.

```
typedef
[switch_type(DWORD)]
union _SHARE_ENUM_UNION {
    [case(0)]
        SHARE_INFO_0_CONTAINER* Level0;
    [case(1)]
        SHARE_INFO_1_CONTAINER* Level1;
    [case(2)]
        SHARE_INFO_2_CONTAINER* Level2;
    [case(501)]
        SHARE_INFO_501_CONTAINER* Level501;
    [case(502)]
        SHARE_INFO_502_CONTAINER* Level502;
    [case(503)]
        SHARE_INFO_503_CONTAINER* Level503;
} SHARE_ENUM_UNION;
```

**Level0:** A pointer to a structure that contains information about shares, as specified in section [2.2.4.32](#).

**Level1:** A pointer to a structure that contains information about shares, as specified in section [2.2.4.33](#).

**Level2:** A pointer to a structure that contains information about shares, as specified in section [2.2.4.34](#).

**Level501:** A pointer to a structure that contains information about shares, as specified in section [2.2.4.35](#).

**Level502:** A pointer to a structure that contains information about shares, as specified in section [2.2.4.36](#).

**Level503:** A pointer to a structure that contains information about shares, as specified in section [2.2.4.37](#).

## 2.2.3.6 SHARE\_INFO

Article02/14/2019

The SHARE\_INFO union contains information about a share.

```
typedef
[switch_type(unsigned long)]
union _SHARE_INFO {
    [case(0)]
        LPSHARE_INFO_0 ShareInfo0;
    [case(1)]
        LPSHARE_INFO_1 ShareInfo1;
    [case(2)]
        LPSHARE_INFO_2 ShareInfo2;
    [case(502)]
        LPSHARE_INFO_502_I ShareInfo502;
    [case(1004)]
        LPSHARE_INFO_1004 ShareInfo1004;
    [case(1006)]
        LPSHARE_INFO_1006 ShareInfo1006;
    [case(1501)]
        LPSHARE_INFO_1501_I ShareInfo1501;
    [default]
    [case(1005)]
        LPSHARE_INFO_1005 ShareInfo1005;
    [case(501)]
        LPSHARE_INFO_501 ShareInfo501;
    [case(503)]
        LPSHARE_INFO_503_I ShareInfo503;
} SHARE_INFO,
*PSHARE_INFO,
*LPSHARE_INFO;
```

**ShareInfo0:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.22](#).

**ShareInfo1:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.23](#).

**ShareInfo2:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.24](#).

**ShareInfo502:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.26](#).

**ShareInfo1004:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.28](#).

**ShareInfo1006:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.30](#).

**ShareInfo1501:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.31](#).

**ShareInfo1005:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.29](#).

**ShareInfo501:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.25](#).

**ShareInfo503:** A pointer to a structure that contains information about a share, as specified in section [2.2.4.27](#).

## 2.2.3.7 SERVER\_INFO

Article10/04/2021

The SERVER\_INFO union contains information about a [server](#).

```
typedef
[switch_type(unsigned long)]
union _SERVER_INFO {
    [case(100)]
        LPSERVER_INFO_100 ServerInfo100;
    [case(101)]
        LPSERVER_INFO_101 ServerInfo101;
    [case(102)]
        LPSERVER_INFO_102 ServerInfo102;
    [case(103)]
        LPSERVER_INFO_103 ServerInfo103;
    [case(502)]
        LPSERVER_INFO_502 ServerInfo502;
    [case(503)]
        LPSERVER_INFO_503 ServerInfo503;
    [case(599)]
        LPSERVER_INFO_599 ServerInfo599;
    [case(1005)]
        LPSERVER_INFO_1005 ServerInfo1005;
    [case(1107)]
        LPSERVER_INFO_1107 ServerInfo1107;
    [case(1010)]
        LPSERVER_INFO_1010 ServerInfo1010;
    [case(1016)]
        LPSERVER_INFO_1016 ServerInfo1016;
    [case(1017)]
        LPSERVER_INFO_1017 ServerInfo1017;
    [case(1018)]
        LPSERVER_INFO_1018 ServerInfo1018;
    [case(1501)]
        LPSERVER_INFO_1501 ServerInfo1501;
    [case(1502)]
        LPSERVER_INFO_1502 ServerInfo1502;
    [case(1503)]
        LPSERVER_INFO_1503 ServerInfo1503;
    [case(1506)]
        LPSERVER_INFO_1506 ServerInfo1506;
    [case(1510)]
        LPSERVER_INFO_1510 ServerInfo1510;
    [case(1511)]
        LPSERVER_INFO_1511 ServerInfo1511;
    [case(1512)]
        LPSERVER_INFO_1512 ServerInfo1512;
    [case(1513)]
        LPSERVER_INFO_1513 ServerInfo1513;
    [case(1514)]
```

```
    LPSERVER_INFO_1514 ServerInfo1514;
[case(1515)]
    LPSERVER_INFO_1515 ServerInfo1515;
[case(1516)]
    LPSERVER_INFO_1516 ServerInfo1516;
[case(1518)]
    LPSERVER_INFO_1518 ServerInfo1518;
[case(1523)]
    LPSERVER_INFO_1523 ServerInfo1523;
[case(1528)]
    LPSERVER_INFO_1528 ServerInfo1528;
[case(1529)]
    LPSERVER_INFO_1529 ServerInfo1529;
[case(1530)]
    LPSERVER_INFO_1530 ServerInfo1530;
[case(1533)]
    LPSERVER_INFO_1533 ServerInfo1533;
[case(1534)]
    LPSERVER_INFO_1534 ServerInfo1534;
[case(1535)]
    LPSERVER_INFO_1535 ServerInfo1535;
[case(1536)]
    LPSERVER_INFO_1536 ServerInfo1536;
[case(1538)]
    LPSERVER_INFO_1538 ServerInfo1538;
[case(1539)]
    LPSERVER_INFO_1539 ServerInfo1539;
[case(1540)]
    LPSERVER_INFO_1540 ServerInfo1540;
[case(1541)]
    LPSERVER_INFO_1541 ServerInfo1541;
[case(1542)]
    LPSERVER_INFO_1542 ServerInfo1542;
[case(1543)]
    LPSERVER_INFO_1543 ServerInfo1543;
[case(1544)]
    LPSERVER_INFO_1544 ServerInfo1544;
[case(1545)]
    LPSERVER_INFO_1545 ServerInfo1545;
[case(1546)]
    LPSERVER_INFO_1546 ServerInfo1546;
[case(1547)]
    LPSERVER_INFO_1547 ServerInfo1547;
[case(1548)]
    LPSERVER_INFO_1548 ServerInfo1548;
[case(1549)]
    LPSERVER_INFO_1549 ServerInfo1549;
[case(1550)]
    LPSERVER_INFO_1550 ServerInfo1550;
[case(1552)]
    LPSERVER_INFO_1552 ServerInfo1552;
[case(1553)]
    LPSERVER_INFO_1553 ServerInfo1553;
[case(1554)]
    LPSERVER_INFO_1554 ServerInfo1554;
```

```
[case(1555)]
    LP SERVER_INFO_1555 ServerInfo1555;
[case(1556)]
    LP SERVER_INFO_1556 ServerInfo1556;
} SERVER_INFO,
*PSERVER_INFO,
*LPSERVER_INFO;
```

**ServerInfo100:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.40](#).

**ServerInfo101:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.41](#).

**ServerInfo102:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.42](#).

**ServerInfo103:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.43.<8>](#)

**ServerInfo502:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.44](#).

**ServerInfo503:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.45](#).

**ServerInfo599:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.46](#).

**ServerInfo1005:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.47](#).

**ServerInfo1107:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.48](#).

**ServerInfo1010:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.49](#).

**ServerInfo1016:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.50](#).

**ServerInfo1017:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.51](#).

**ServerInfo1018:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.52](#).

**ServerInfo1501:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.53](#).

**ServerInfo1502:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.54](#).

**ServerInfo1503:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.55](#).

**ServerInfo1506:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.56](#).

**ServerInfo1510:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.57](#).

**ServerInfo1511:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.58](#).

**ServerInfo1512:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.59](#).

**ServerInfo1513:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.60](#).

**ServerInfo1514:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.61](#).

**ServerInfo1515:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.62](#)

**ServerInfo1516:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.63](#).

**ServerInfo1518:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.64](#).

**ServerInfo1523:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.65](#).

**ServerInfo1528:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.66](#).

**ServerInfo1529:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.67](#).

**ServerInfo1530:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.68](#).

**ServerInfo1533:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.69](#).

**ServerInfo1534:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.70](#).

**ServerInfo1535:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.71](#).

**ServerInfo1536:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.72](#).

**ServerInfo1538:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.73](#).

**ServerInfo1539:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.74](#).

**ServerInfo1540:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.75](#).

**ServerInfo1541:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.76](#).

**ServerInfo1542:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.77](#).

**ServerInfo1543:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.78](#).

**ServerInfo1544:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.79](#).

**ServerInfo1545:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.80](#).

**ServerInfo1546:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.81](#).

**ServerInfo1547:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.82](#).

**ServerInfo1548:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.83](#).

**ServerInfo1549:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.84](#).

**ServerInfo1550:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.85](#).

**ServerInfo1552:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.86](#).

**ServerInfo1553:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.87](#).

**ServerInfo1554:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.88](#).

**ServerInfo1555:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.89](#).

**ServerInfo1556:** A pointer to a structure that contains information about a server, as specified in section [2.2.4.90](#).

## 2.2.3.8 SERVER\_XPORT\_ENUM\_UNION

Article02/14/2019

The SERVER\_XPORT\_ENUM\_UNION union contains information about file [server transports](#).

```
typedef
[switch_type(DWORD)]
union _SERVER_XPORT_ENUM_UNION {
    [case(0)]
        PSERVER_XPORT_INFO_0_CONTAINER Level0;
    [case(1)]
        PSERVER_XPORT_INFO_1_CONTAINER Level1;
    [case(2)]
        PSERVER_XPORT_INFO_2_CONTAINER Level2;
    [case(3)]
        PSERVER_XPORT_INFO_3_CONTAINER Level3;
} SERVER_XPORT_ENUM_UNION;
```

**Level0:** A pointer to a structure containing information about file server transports, as specified in section [2.2.4.97](#).

**Level1:** A pointer to a structure containing information about file server transports, as specified in section [2.2.4.98](#).

**Level2:** A pointer to a structure containing information about file server transports, as specified in section [2.2.4.99](#).

**Level3:** A pointer to a structure containing information about file server transports, as specified in section [2.2.4.100](#).

## 2.2.3.9 TRANSPORT\_INFO

Article04/06/2021

The TRANSPORT\_INFO union contains information about a transport over which a file server is operational.

```
typedef
[switch_type(unsigned long)]
union _TRANSPORT_INFO {
    [case(0)]
        SERVER_TRANSPORT_INFO_0 Transport0;
    [case(1)]
        SERVER_TRANSPORT_INFO_1 Transport1;
    [case(2)]
        SERVER_TRANSPORT_INFO_2 Transport2;
    [case(3)]
        SERVER_TRANSPORT_INFO_3 Transport3;
} TRANSPORT_INFO,
*PTRANSPORT_INFO,
*LPTRANSPORT_INFO;
```

**Transport0:** A pointer to a structure containing information about a file server transport, as specified in section [2.2.4.93](#).

**Transport1:** A pointer to a structure containing information about a file server transport, as specified in section [2.2.4.94](#).

**Transport2:** A pointer to a structure containing information about a file server transport, as specified in section [2.2.4.95](#).

**Transport3:** A pointer to a structure containing information about a file server transport, as specified in section [2.2.4.96](#).

## 2.2.3.10 SERVER\_ALIAS\_INFO

Article02/14/2019

The SERVER\_ALIAS\_INFO union contains information about an alias attached to a [server](#) name.

```
typedef
[switch_type(unsigned long)]
union _SERVER_ALIAS_INFO {
    [case(0)]
        LPSERVER_ALIAS_INFO_0 ServerAliasInfo0;
} SERVER_ALIAS_INFO,
*PSERVER_ALIAS_INFO,
*LPSERVER_ALIAS_INFO;
```

**ServerAliasInfo0:** A pointer to a structure containing information about an alias attached to a server, as specified in section [2.2.4.102](#).

## 2.2.4 Structures

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [2.2.4.1 CONNECTION\\_INFO\\_0](#)
- [2.2.4.2 CONNECTION\\_INFO\\_1](#)
- [2.2.4.3 CONNECT\\_INFO\\_0\\_CONTAINER](#)
- [2.2.4.4 CONNECT\\_INFO\\_1\\_CONTAINER](#)
- [2.2.4.5 CONNECT\\_ENUM\\_STRUCT](#)
- [2.2.4.6 FILE\\_INFO\\_2](#)
- [2.2.4.7 FILE\\_INFO\\_3](#)
- [2.2.4.8 FILE\\_INFO\\_2\\_CONTAINER](#)
- [2.2.4.9 FILE\\_INFO\\_3\\_CONTAINER](#)
- [2.2.4.10 FILE\\_ENUM\\_STRUCT](#)
- [2.2.4.11 SESSION\\_INFO\\_0](#)
- [2.2.4.12 SESSION\\_INFO\\_1](#)
- [2.2.4.13 SESSION\\_INFO\\_2](#)
- [2.2.4.14 SESSION\\_INFO\\_10](#)
- [2.2.4.15 SESSION\\_INFO\\_502](#)
- [2.2.4.16 SESSION\\_INFO\\_0\\_CONTAINER](#)
- [2.2.4.17 SESSION\\_INFO\\_1\\_CONTAINER](#)
- [2.2.4.18 SESSION\\_INFO\\_2\\_CONTAINER](#)
- [2.2.4.19 SESSION\\_INFO\\_10\\_CONTAINER](#)
- [2.2.4.20 SESSION\\_INFO\\_502\\_CONTAINER](#)
- [2.2.4.21 SESSION\\_ENUM\\_STRUCT](#)
- [2.2.4.22 SHARE\\_INFO\\_0](#)
- [2.2.4.23 SHARE\\_INFO\\_1](#)
- [2.2.4.24 SHARE\\_INFO\\_2](#)
- [2.2.4.25 SHARE\\_INFO\\_501](#)
- [2.2.4.26 SHARE\\_INFO\\_502\\_I](#)
- [2.2.4.27 SHARE\\_INFO\\_503\\_I](#)
- [2.2.4.28 SHARE\\_INFO\\_1004](#)
- [2.2.4.29 SHARE\\_INFO\\_1005](#)
- [2.2.4.30 SHARE\\_INFO\\_1006](#)
- [2.2.4.31 SHARE\\_INFO\\_1501\\_I](#)
- [2.2.4.32 SHARE\\_INFO\\_0\\_CONTAINER](#)
- [2.2.4.33 SHARE\\_INFO\\_1\\_CONTAINER](#)
- [2.2.4.34 SHARE\\_INFO\\_2\\_CONTAINER](#)

- 2.2.4.35 SHARE\_INFO\_501\_CONTAINER
- 2.2.4.36 SHARE\_INFO\_502\_CONTAINER
- 2.2.4.37 SHARE\_INFO\_503\_CONTAINER
- 2.2.4.38 SHARE\_ENUM\_STRUCT
- 2.2.4.39 STAT\_SERVER\_0
- 2.2.4.40 SERVER\_INFO\_100
- 2.2.4.41 SERVER\_INFO\_101
- 2.2.4.42 SERVER\_INFO\_102
- 2.2.4.43 SERVER\_INFO\_103
- 2.2.4.44 SERVER\_INFO\_502
- 2.2.4.45 SERVER\_INFO\_503
- 2.2.4.46 SERVER\_INFO\_599
- 2.2.4.47 SERVER\_INFO\_1005
- 2.2.4.48 SERVER\_INFO\_1107
- 2.2.4.49 SERVER\_INFO\_1010
- 2.2.4.50 SERVER\_INFO\_1016
- 2.2.4.51 SERVER\_INFO\_1017
- 2.2.4.52 SERVER\_INFO\_1018
- 2.2.4.53 SERVER\_INFO\_1501
- 2.2.4.54 SERVER\_INFO\_1502
- 2.2.4.55 SERVER\_INFO\_1503
- 2.2.4.56 SERVER\_INFO\_1506
- 2.2.4.57 SERVER\_INFO\_1510
- 2.2.4.58 SERVER\_INFO\_1511
- 2.2.4.59 SERVER\_INFO\_1512
- 2.2.4.60 SERVER\_INFO\_1513
- 2.2.4.61 SERVER\_INFO\_1514
- 2.2.4.62 SERVER\_INFO\_1515
- 2.2.4.63 SERVER\_INFO\_1516
- 2.2.4.64 SERVER\_INFO\_1518
- 2.2.4.65 SERVER\_INFO\_1523
- 2.2.4.66 SERVER\_INFO\_1528
- 2.2.4.67 SERVER\_INFO\_1529
- 2.2.4.68 SERVER\_INFO\_1530
- 2.2.4.69 SERVER\_INFO\_1533
- 2.2.4.70 SERVER\_INFO\_1534
- 2.2.4.71 SERVER\_INFO\_1535
- 2.2.4.72 SERVER\_INFO\_1536
- 2.2.4.73 SERVER\_INFO\_1538
- 2.2.4.74 SERVER\_INFO\_1539

- 2.2.4.75 SERVER\_INFO\_1540
- 2.2.4.76 SERVER\_INFO\_1541
- 2.2.4.77 SERVER\_INFO\_1542
- 2.2.4.78 SERVER\_INFO\_1543
- 2.2.4.79 SERVER\_INFO\_1544
- 2.2.4.80 SERVER\_INFO\_1545
- 2.2.4.81 SERVER\_INFO\_1546
- 2.2.4.82 SERVER\_INFO\_1547
- 2.2.4.83 SERVER\_INFO\_1548
- 2.2.4.84 SERVER\_INFO\_1549
- 2.2.4.85 SERVER\_INFO\_1550
- 2.2.4.86 SERVER\_INFO\_1552
- 2.2.4.87 SERVER\_INFO\_1553
- 2.2.4.88 SERVER\_INFO\_1554
- 2.2.4.89 SERVER\_INFO\_1555
- 2.2.4.90 SERVER\_INFO\_1556
- 2.2.4.91 DISK\_INFO
- 2.2.4.92 DISK\_ENUM\_CONTAINER
- 2.2.4.93 SERVER\_TRANSPORT\_INFO\_0
- 2.2.4.94 SERVER\_TRANSPORT\_INFO\_1
- 2.2.4.95 SERVER\_TRANSPORT\_INFO\_2
- 2.2.4.96 SERVER\_TRANSPORT\_INFO\_3
- 2.2.4.97 SERVER\_XPORT\_INFO\_0\_CONTAINER
- 2.2.4.98 SERVER\_XPORT\_INFO\_1\_CONTAINER
- 2.2.4.99 SERVER\_XPORT\_INFO\_2\_CONTAINER
- 2.2.4.100 SERVER\_XPORT\_INFO\_3\_CONTAINER
- 2.2.4.101 SERVER\_XPORT\_ENUM\_STRUCT
- 2.2.4.102 SERVER\_ALIAS\_INFO\_0
- 2.2.4.103 SERVER\_ALIAS\_INFO\_0\_CONTAINER
- 2.2.4.104 SERVER\_ALIAS\_ENUM\_STRUCT
- 2.2.4.105 TIME\_OF\_DAY\_INFO
- 2.2.4.106 ADT\_SECURITY\_DESCRIPTOR
- 2.2.4.107 NET\_DFS\_ENTRY\_ID
- 2.2.4.108 NET\_DFS\_ENTRY\_ID\_CONTAINER
- 2.2.4.109 DFS\_SITENAME\_INFO
- 2.2.4.110 DFS\_SITELIST\_INFO

## 2.2.4.1 CONNECTION\_INFO\_0

Article02/14/2019

The CONNECTION\_INFO\_0 structure contains the identifier of a connection.

```
typedef struct _CONNECTION_INFO_0 {
    DWORD coni0_id;
} CONNECTION_INFO_0,
*PCONNECTION_INFO_0,
*LPCONNECTION_INFO_0;
```

**coni0\_id:** Specifies a connection identifier. For more information, see [Abstract Data Model \(section 3.1.1\)](#).

## 2.2.4.2 CONNECTION\_INFO\_1

Article 06/24/2021

The CONNECTION\_INFO\_1 structure contains the identifier of a [connection](#), the number of open files, the connection time, the number of users on the connection, and the type of connection.

```
typedef struct _CONNECTION_INFO_1 {
    DWORD coni1_id;
    DWORD coni1_type;
    DWORD coni1_numOpens;
    DWORD coni1_numUsers;
    DWORD coni1_time;
    [string] wchar_t* coni1_username;
    [string] wchar_t* coni1_netname;
} CONNECTION_INFO_1,
*PCONNECTION_INFO_1,
*LPCONNECTION_INFO_1;
```

**coni1\_id:** Specifies a connection identifier.

**coni1\_type:** Specifies the type of connection made from the local device name to the shared resource. It MUST be one of the values listed in section [2.2.2.4](#).

**coni1\_numOpens:** Specifies the number of files that are currently opened by using the connection.

**coni1\_numUsers:** Specifies the number of users on the connection.

**coni1\_time:** Specifies the number of seconds that the connection has been established.

**coni1\_username:** A pointer to a null-terminated Unicode UTF-16 string that specifies the name of the user that is associated with the connection.

**coni1\_netname:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) which is the computer name of the [client](#). The value of this member depends on which name was specified as the *Qualifier* parameter to the [NetrConnectionEnum \(section 3.1.4.1\)](#) method. The name that is not specified in the *Qualifier* parameter to NetrConnectionEnum MUST be returned in the coni1\_netname field.

## 2.2.4.3 CONNECT\_INFO\_0\_CONTAINER

Article02/14/2019

The CONNECT\_INFO\_0\_CONTAINER structure contains a value that indicates the number of entries that the [NetrConnectionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _CONNECT_INFO_0_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPCONNECTION_INFO_0 Buffer;
} CONNECT_INFO_0_CONTAINER,
*PCONNECT_INFO_0_CONTAINER,
*LPCONNECT_INFO_0_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [CONNECTION\\_INFO\\_0](#) entries returned by the method.

## 2.2.4.4 CONNECT\_INFO\_1\_CONTAINER

Article02/14/2019

The CONNECT\_INFO\_1\_CONTAINER structure contains a value that indicates the number of entries that the [NetrConnectionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _CONNECT_INFO_1_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPCONNECTION_INFO_1 Buffer;
} CONNECT_INFO_1_CONTAINER,
*PCONNECT_INFO_1_CONTAINER,
*LPCONNECT_INFO_1_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [CONNECTION\\_INFO\\_1](#) entries returned by the method.

## 2.2.4.5 CONNECT\_ENUM\_STRUCT

Article 10/30/2020

The CONNECT\_ENUM\_STRUCT structure specifies the information level that the [client](#) requests when invoking the [NetrConnectionEnum](#) (section 3.1.4.1) method and encapsulates the [CONNECT\\_ENUM\\_UNION](#) (section 2.2.3.1) union that receives the entries that are enumerated by the [server](#).

```
typedef struct _CONNECT_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] CONNECT_ENUM_UNION ConnectInfo;
} CONNECT_ENUM_STRUCT,
*PCONNECT_ENUM_STRUCT,
*LPCONNECT_ENUM_STRUCT;
```

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
0	CONNECT_INFO_0_CONTAINER
1	CONNECT_INFO_1_CONTAINER

**ConnectInfo:** Contains either a [CONNECT\\_INFO\\_0\\_CONTAINER](#) structure or a [CONNECT\\_INFO\\_1\\_CONTAINER](#) structure depending on the value of the *Level* parameter. The enumerated elements are returned in this member.

## 2.2.4.6 FILE\_INFO\_2

Article02/14/2019

The FILE\_INFO\_2 structure contains the identifier for a file, device, or pipe.

```
typedef struct _FILE_INFO_2 {
    DWORD fi2_id;
} FILE_INFO_2,
*PFILE_INFO_2,
*LPFILE_INFO_2;
```

**fi2\_id:** Specifies a DWORD value that contains the identifier that is assigned to the file, device, or pipe when it was opened. See section [3.1.1](#) for details.

## 2.2.4.7 FILE\_INFO\_3

Article10/30/2020

The FILE\_INFO\_3 structure contains the identifier and other pertinent information about files, devices, and pipes.

```
typedef struct _FILE_INFO_3 {
    DWORD fi3_id;
    DWORD fi3_permissions;
    DWORD fi3_num_locks;
    [string] wchar_t* fi3_pathname;
    [string] wchar_t* fi3_username;
} FILE_INFO_3,
*PFILE_INFO_3,
*LPFILE_INFO_3;
```

**fi3\_id:** Specifies a DWORD value that contains the identifier that is assigned to the file, device, or pipe when it was opened. See section [3.1.1](#) for details.

**fi3\_permissions:** Specifies a DWORD value that contains the access permissions that are associated with the opening application. This member MUST be a combination of one or more of the following values.

Value	Meaning
PERM_FILE_READ 0x00000001	Permission to read a resource, and, by default, execute the resource.
PERM_FILE_WRITE 0x00000002	Permission to write to a resource.
PERM_FILE_CREATE 0x00000004	Permission to create a resource; data can be written when creating the resource.
ACCESS_EXEC 0x00000008	Permission to execute a resource.
ACCESS_DELETE 0x00000010	Permission to delete a resource.
ACCESS_ATTRIB 0x00000020	Permission to modify the attributes of a resource.

Value	Meaning
ACCESS_PERM 0x00000040	Permission to modify the permissions assigned to a resource for a user or application.

**fi3\_num\_locks:** Specifies a DWORD value that contains the number of file locks on the file, device, or pipe.

**fi3\_pathname:** A pointer to a string that specifies the path of the opened file, device, or pipe.

**fi3\_username:** A pointer to a string that specifies which user opened the file, device, or pipe.

## 2.2.4.8 FILE\_INFO\_2\_CONTAINER

Article02/14/2019

The FILE\_INFO\_2\_CONTAINER structure contains a value that indicates the number of entries that the [NetrFileEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _FILE_INFO_2_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPFILE_INFO_2 Buffer;
} FILE_INFO_2_CONTAINER,
*PFILE_INFO_2_CONTAINER,
*LPFILE_INFO_2_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [FILE\\_INFO\\_2](#) entries returned by the method.

## 2.2.4.9 FILE\_INFO\_3\_CONTAINER

Article02/14/2019

The FILE\_INFO\_3\_CONTAINER structure contains a value that indicates the number of entries that the [NetrFileEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _FILE_INFO_3_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPFILE_INFO_3 Buffer;
} FILE_INFO_3_CONTAINER,
*PFILE_INFO_3_CONTAINER,
*LPFILE_INFO_3_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [FILE\\_INFO\\_3](#) entries returned by the method.

## 2.2.4.10 FILE\_ENUM\_STRUCT

Article10/30/2020

The FILE\_ENUM\_STRUCT structure specifies the information level that the [client](#) requests in the [NetrFileEnum](#) method and encapsulates the [FILE\\_ENUM\\_UNION](#) union that receives the entries that are enumerated by the [server](#).

```
typedef struct _FILE_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] FILE_ENUM_UNION FileInfo;
} FILE_ENUM_STRUCT,
*PFILE_ENUM_STRUCT,
*LPFILE_ENUM_STRUCT;
```

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
2	FILE_INFO_2_CONTAINER
3	FILE_INFO_3_CONTAINER

**FileInfo:** Contains a file info container structure whose type is determined by the *Level* parameter as shown in the preceding table. The enumerated elements are returned in this member.

## 2.2.4.11 SESSION\_INFO\_0

Article02/14/2019

The SESSION\_INFO\_0 structure contains the name of the computer that established the session.

```
typedef struct _SESSION_INFO_0 {
    [string] wchar_t* sesi0_cname;
} SESSION_INFO_0,
*PSESSION_INFO_0,
*LPSESSION_INFO_0;
```

**sesi0\_cname:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) of the computer that established the session.

## 2.2.4.12 SESSION\_INFO\_1

Article02/14/2019

The SESSION\_INFO\_1 structure contains information about the session, including the name of the computer and user; open files, pipes, and devices that are on the computer; session active and idle times; and how the user established the session.

```
typedef struct _SESSION_INFO_1 {
    [string] wchar_t* sesi1_cname;
    [string] wchar_t* sesi1_username;
    DWORD sesi1_numOpens;
    DWORD sesi1_time;
    DWORD sesi1_idle_time;
    DWORD sesi1_user_flags;
} SESSION_INFO_1,
*PSESSION_INFO_1,
*LPSESSION_INFO_1;
```

**sesi1\_cname:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) of the computer that established the session.

**sesi1\_username:** A pointer to a null-terminated Unicode UTF-16 string that specifies the name of the user who established the session.

**sesi1\_numOpens:** Specifies a [DWORD](#) value that contains the number of files, devices, and pipes that were opened during the session.

**sesi1\_time:** Specifies a [DWORD](#) value that contains the number of seconds since the session was created.

**sesi1\_idle\_time:** Specifies a [DWORD](#) value that contains the number of seconds the session has been idle.

**sesi1\_user\_flags:** Specifies a [DWORD](#) value that specifies how the user established the session. This member MUST be a combination of one or more of the values that are defined in [2.2.2.3](#).

## 2.2.4.13 SESSION\_INFO\_2

Article02/14/2019

The SESSION\_INFO\_2 structure contains information about the session, including the name of the computer; name of the user; open files, pipes, and devices that are on the computer; session active and idle times; how the user established the session; and the type of [client](#) that established the session.

```
typedef struct _SESSION_INFO_2 {
    [string] wchar_t* sesi2_cname;
    [string] wchar_t* sesi2_username;
    DWORD sesi2_numOpens;
    DWORD sesi2_time;
    DWORD sesi2_idle_time;
    DWORD sesi2_user_flags;
    [string] wchar_t* sesi2_cltype_name;
} SESSION_INFO_2,
*PSESSION_INFO_2,
*LPSESSION_INFO_2;
```

**sesi2\_cname:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) of the computer that established the session.

**sesi2\_username:** A pointer to a null-terminated Unicode UTF-16 string that specifies the name of the user who established the session.

**sesi2\_numOpens:** Specifies a DWORD value that contains the number of files, devices, and pipes that were opened during the session.

**sesi2\_time:** Specifies a DWORD value that contains the number of seconds the session has been active.

**sesi2\_idle\_time:** Specifies a DWORD value that contains the number of seconds the session has been idle.

**sesi2\_user\_flags:** Specifies a DWORD value that describes how the user established the session. This member MUST be a combination of one or more of the values that are defined in section [2.2.2.3](#).

**sesi2\_cltype\_name:** A pointer to a null-terminated Unicode UTF-16 string that specifies the type of client that established the session. The [server](#) simply stores this string, as specified in section [2.2.2.1](#), and its value does not modify the behavior of the protocol.

<9>

## 2.2.4.14 SESSION\_INFO\_10

Article02/14/2019

The SESSION\_INFO\_10 structure contains information about the session, including the name of the computer, the name of the user, and the active and idle times for the session.

```
typedef struct _SESSION_INFO_10 {
    [string] wchar_t* sesi10_cname;
    [string] wchar_t* sesi10_username;
    DWORD sesi10_time;
    DWORD sesi10_idle_time;
} SESSION_INFO_10,
*PSESSION_INFO_10,
*LPSESSION_INFO_10;
```

**sesi10\_cname:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) of the computer that established the session.

**sesi10\_username:** A pointer to a null-terminated Unicode UTF-16 string specifying the name of the user who established the session.

**sesi10\_time:** Specifies the number of seconds the session has been active.

**sesi10\_idle\_time:** Specifies the number of seconds the session has been idle.

## 2.2.4.15 SESSION\_INFO\_502

Article02/14/2019

The SESSION\_INFO\_502 structure contains information about the session, including the name of the computer; the name of the user; open files, pipes, and devices that are on the computer; the [client](#) type; session active and idle times; how the user established the session; and the name of the transport that the client is using.

```
typedef struct _SESSION_INFO_502 {
    [string] wchar_t* sesi502_cname;
    [string] wchar_t* sesi502_username;
    DWORD sesi502_numOpens;
    DWORD sesi502_time;
    DWORD sesi502_idle_time;
    DWORD sesi502_user_flags;
    [string] wchar_t* sesi502_cltype_name;
    [string] wchar_t* sesi502_transport;
} SESSION_INFO_502,
*PSESSION_INFO_502,
*LPSESSION_INFO_502;
```

**sesi502\_cname:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) of the computer that established the session.

**sesi502\_username:** A pointer to a null-terminated Unicode UTF-16 string that specifies the name of the user who established the session.

**sesi502\_numOpens:** Specifies the number of files, devices, and pipes that were opened during the session.

**sesi502\_time:** Specifies the number of seconds the session has been active.

**sesi502\_idle\_time:** Specifies the number of seconds the session has been idle.

**sesi502\_user\_flags:** Specifies a value that describes how the user established the session. This member MUST be a combination of one or more of the values that are listed in section [2.2.2.3](#).

**sesi502\_cltype\_name:** A pointer to a null-terminated Unicode UTF-16 string that specifies the type of client that established the session. The [server](#) simply stores this string, as specified in section [2.2.2.1](#), and its value does not modify the behavior of the protocol.<10>

**sesi502\_transport:** Specifies the name of the transport that the client is using to communicate with the server.

## 2.2.4.16 SESSION\_INFO\_0\_CONTAINER

Article02/14/2019

The SESSION\_INFO\_0\_CONTAINER structure contains a value that indicates the number of entries that the [NetrSessionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SESSION_INFO_0_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_0 Buffer;
} SESSION_INFO_0_CONTAINER,
*PSESSION_INFO_0_CONTAINER,
*LPSESSION_INFO_0_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SESSION\\_INFO\\_0](#) entries returned by the method.

## 2.2.4.17 SESSION\_INFO\_1\_CONTAINER

Article 06/24/2021

The SESSION\_INFO\_1\_CONTAINER structure contains a value that indicates the number of entries that the [NetrSessionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SESSION_INFO_1_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_1 Buffer;
} SESSION_INFO_1_CONTAINER,
*PSESSION_INFO_1_CONTAINER,
*LPSESSION_INFO_1_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SESSION\\_INFO\\_1](#) entries returned by the method.

## 2.2.4.18 SESSION\_INFO\_2\_CONTAINER

Article02/14/2019

The SESSION\_INFO\_2\_CONTAINER structure contains a value that indicates the number of entries that the [NetrSessionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SESSION_INFO_2_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_2 Buffer;
} SESSION_INFO_2_CONTAINER,
*PSESSION_INFO_2_CONTAINER,
*LPSESSION_INFO_2_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SESSION\\_INFO\\_2](#) entries returned by the method.

## 2.2.4.19 SESSION\_INFO\_10\_CONTAINER

Article02/14/2019

The SESSION\_INFO\_10\_CONTAINER structure contains a value that indicates the number of entries that the [NetrSessionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SESSION_INFO_10_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_10 Buffer;
} SESSION_INFO_10_CONTAINER,
*PSESSION_INFO_10_CONTAINER,
*LPSESSION_INFO_10_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SESSION\\_INFO\\_10](#) entries returned by the method.

## 2.2.4.20

# SESSION\_INFO\_502\_CONTAINER

Article02/14/2019

The SESSION\_INFO\_502\_CONTAINER structure contains a value that indicates the number of entries that the [NetrSessionEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SESSION_INFO_502_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_502 Buffer;
} SESSION_INFO_502_CONTAINER,
*PSESSION_INFO_502_CONTAINER,
*LPSESSION_INFO_502_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SESSION\\_INFO\\_502](#) entries returned by the method.

## 2.2.4.21 SESSION\_ENUM\_STRUCT

Article10/30/2020

The SESSION\_ENUM\_STRUCT structure specifies the information level that the [client](#) requests in the [NetrSessionEnum](#) method and encapsulates the [SESSION\\_ENUM\\_UNION](#) union that receives the entries that are enumerated by the [server](#).

```
typedef struct _SESSION_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] SESSION_ENUM_UNION SessionInfo;
} SESSION_ENUM_STRUCT,
*PSESSION_ENUM_STRUCT,
*LPSESSION_ENUM_STRUCT;
```

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
0	SESSION_INFO_0_CONTAINER
1	SESSION_INFO_1_CONTAINER
2	SESSION_INFO_2_CONTAINER
10	SESSION_INFO_10_CONTAINER
502	SESSION_INFO_502_CONTAINER

**SessionInfo:** Contains a session info container whose type is specified by the *Level* parameter, as shown in the preceding table. The enumerated session entries are returned in this member.

## 2.2.4.22 SHARE\_INFO\_0

Article02/14/2019

The SHARE\_INFO\_0 structure contains the name of the shared resource. For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_I](#) ([section 2.2.4.26](#)) structure (shi0\_xxx denotes the same information as shi502\_xxx).

```
typedef struct _SHARE_INFO_0 {
    [string] wchar_t* shi0_netname;
} SHARE_INFO_0,
*PSHARE_INFO_0,
*LPSHARE_INFO_0;
```

## 2.2.4.23 SHARE\_INFO\_1

Article02/14/2019

The SHARE\_INFO\_1 structure contains information about the shared resource, including the name and type of the resource and a comment associated with the resource. For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_I \(section 2.2.4.26\)](#) structure (shi1\_xxx denotes the same information as shi502\_xxx).

```
typedef struct _SHARE_INFO_1 {
    [string] wchar_t* shi1_netname;
    DWORD shi1_type;
    [string] wchar_t* shi1_remark;
} SHARE_INFO_1,
*PSHARE_INFO_1,
*LPSHARE_INFO_1;
```

## 2.2.4.24 SHARE\_INFO\_2

Article02/14/2019

The SHARE\_INFO\_2 structure contains information about the shared resource, including the name, type, and permissions of the resource, comments associated with the resource, the maximum number of concurrent [connections](#), the number of current connections, the local path for the resource, and a password for the current connection. For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_1 \(section 2.2.4.26\)](#) structure (shi2\_xxx denotes the same information as shi502\_xxx).

```
typedef struct _SHARE_INFO_2 {
    [string] wchar_t* shi2_netname;
    DWORD shi2_type;
    [string] wchar_t* shi2_remark;
    DWORD shi2_permissions;
    DWORD shi2_max_uses;
    DWORD shi2_current_uses;
    [string] wchar_t* shi2_path;
    [string] wchar_t* shi2_passwd;
} SHARE_INFO_2,
*PSHARE_INFO_2,
*LPSHARE_INFO_2;
```

## 2.2.4.25 SHARE\_INFO\_501

Article02/14/2019

The SHARE\_INFO\_501 structure contains information about the shared resource, including the name and type of the resource and a comment that is associated with the resource. For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_1 \(section 2.2.4.26\)](#) structure (shi501\_netname, shi501\_type, and shi501\_remark denote the same information as shi502\_xxx in section 2.2.4.26, and shi501\_flags denotes the same information as shi1005\_flags in section [2.2.4.29](#)).

```
typedef struct _SHARE_INFO_501 {
    [string] wchar_t* shi501_netname;
    DWORD shi501_type;
    [string] wchar_t* shi501_remark;
    DWORD shi501_flags;
} SHARE_INFO_501,
*PSHARE_INFO_501,
*LPSHARE_INFO_501;
```

## 2.2.4.26 SHARE\_INFO\_502\_I

Article 06/24/2021

The SHARE\_INFO\_502\_I structure contains information about the shared resource, including the name of the resource, type, and permissions, the number of [connections](#), and other pertinent information.

```
typedef struct _SHARE_INFO_502_I {
    [string] WCHAR* shi502_netname;
    DWORD shi502_type;
    [string] WCHAR* shi502_remark;
    DWORD shi502_permissions;
    DWORD shi502_max_uses;
    DWORD shi502_current_uses;
    [string] WCHAR* shi502_path;
    [string] WCHAR* shi502_passwd;
    DWORD shi502_reserved;
    [size_is(shi502_reserved)] unsigned char* shi502_security_descriptor;
} SHARE_INFO_502_I,
*PSHARE_INFO_502_I,
*LPSHARE_INFO_502_I;
```

**shi502\_netname:** A pointer to a null-terminated Unicode UTF-16 string that specifies the name of a shared resource. The [server](#) MUST ignore this member when processing the [NetrShareSetInfo](#) ([section 3.1.4.11](#)) method.

**shi502\_type:** Specifies a DWORD value that indicates the type of share. The server MUST ignore this member when processing the NetrShareSetInfo method; otherwise, it MUST be one of the values that are listed in section [2.2.2.4](#).

**shi502\_remark:** A pointer to a null-terminated Unicode UTF-16 string that specifies an optional comment about the shared resource.

**shi502\_permissions:** This field is not used. The server MUST ignore the value of this parameter on receipt.

**shi502\_max\_uses:** Specifies a DWORD value that indicates the maximum number of concurrent connections that the shared resource can accommodate. If the value that is specified by **shi502\_max\_uses** is 0xFFFFFFFF, the maximum number of connections MUST be unlimited.

**shi502\_current\_uses:** Specifies a DWORD value that indicates the number of current connections to the resource. The server MUST ignore this member on receipt.

**shi502\_path**: A pointer to a null-terminated Unicode UTF-16 string that contains the local path for the shared resource. For disks, **shi502\_path** is the path that is being shared. For print queues, **shi502\_path** is the name of the print queue that is being shared. For communication devices, **shi502\_path** is the name of the communication device that is being shared. For interprocess communications (IPC), **shi502\_path** is the name of the interprocess communication that is being shared. The server MUST ignore this member when processing the NetrShareSetInfo method.

**shi502\_passwd**: This field is not used. The [client](#) MUST send a NULL (zero-length) string and the server MUST ignore the value of this parameter on receipt.

**shi502\_reserved**: The length of the security descriptor that is being passed in the **shi502\_security\_descriptor** member.

**shi502\_security\_descriptor**: Specifies the SECURITY\_DESCRIPTOR, as described in [\[MS-DTYP\]](#) section 2.4.6, that is associated with this share.

## 2.2.4.27 SHARE\_INFO\_503\_I

Article06/24/2021

The SHARE\_INFO\_503\_I structure contains information about the shared resource, including the name of the resource, type, and permissions, the number of connections, and other pertinent information.

```
typedef struct _SHARE_INFO_503_I {
    [string] WCHAR* shi503_netname;
    DWORD shi503_type;
    [string] WCHAR* shi503_remark;
    DWORD shi503_permissions;
    DWORD shi503_max_uses;
    DWORD shi503_current_uses;
    [string] WCHAR* shi503_path;
    [string] WCHAR* shi503_passwd;
    [string] WCHAR* shi503_servername;
    DWORD shi503_reserved;
    [size_is(shi503_reserved)] PUCHAR shi503_security_descriptor;
} SHARE_INFO_503_I,
*PSHARE_INFO_503_I,
*LPSHARE_INFO_503_I;
```

**shi503\_netname:** A pointer to a null-terminated Unicode UTF-16 string that specifies the name of a shared resource. The server MUST ignore this member when processing the [NetrShareSetInfo \(section 3.1.4.11\)](#) method.

**shi503\_type:** Specifies a [DWORD](#) value that indicates the type of share. The server MUST ignore this member when processing the NetrShareSetInfo method. Otherwise, it MUST be one of the values listed in section [2.2.2.4](#).

**shi503\_remark:** A pointer to a null-terminated Unicode UTF-16 string that specifies an optional comment about the shared resource.

**shi503\_permissions:** This field is not used. The server MUST ignore the value of this parameter on receipt.

**shi503\_max\_uses:** Specifies a DWORD value that indicates the maximum number of concurrent connections that the shared resource can accommodate. If the value is 0xFFFFFFFF, the maximum number of connections MUST be unlimited.

**shi503\_current\_uses:** Specifies a DWORD value that indicates the number of current connections to the resource. The server MUST ignore this member on receipt.

**shi503\_path:** A pointer to a null-terminated Unicode UTF-16 string that contains the local path for the shared resource. For disks, it is the path being shared. For print queues, it is the name of the print queue being shared. The server MUST ignore this member when processing the NetrShareSetInfo method.

**shi503\_passwd:** This field is not used. The client MUST send a NULL (zero-length) string, and the server MUST ignore the value of this parameter on receipt.

**shi503\_servername:** A pointer to a string that specifies the DNS or NetBIOS name of the server on which the shared resource resides. It SHOULD be either "\*" or the string matching one of the server names. Otherwise, the default server name will be used in <shi503\_netname, default server name> to locate a [scoped share](#) as specified in section [2.2.4.102](#). A value of "\*" indicates that there is no configured server name.

**shi503\_reserved:** The length of the security descriptor passed in the **shi503\_security\_descriptor** member.

**shi503\_security\_descriptor:** Specifies the SECURITY\_DESCRIPTOR, as described in [\[MS-DTYP\]](#) section [2.4.6](#), that is associated with this share.

## 2.2.4.28 SHARE\_INFO\_1004

Article02/14/2019

The SHARE\_INFO\_1004 structure contains a comment that is associated with the shared resource. For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_1 \(section 2.2.4.26\)](#) structure (shi1004\_xxx denotes the same information as shi502\_xxx).

```
typedef struct _SHARE_INFO_1004 {
    [string] wchar_t* shi1004_remark;
} SHARE_INFO_1004,
*PSHARE_INFO_1004,
*LPSHARE_INFO_1004;
```

## 2.2.4.29 SHARE\_INFO\_1005

Article04/06/2021

The SHARE\_INFO\_1005 structure contains information about the shared resource.

```
typedef struct _SHARE_INFO_1005 {
    DWORD shi1005_flags;
} SHARE_INFO_1005,
*PSHARE_INFO_1005,
*LPSHARE_INFO_1005;
```

**shi1005\_flags:** Specifies a **DWORD** bitmask value that MUST contain zero or more of the following values. The bit locations that are named CSC\_MASK in the following table MUST contain a client-side caching state value as given in section 2.2.2.5. The [server](#) MUST ignore SHI1005\_FLAGS\_DFS and SHI1005\_FLAGS\_DFS\_ROOT as it processes the [NetrShareSetInfo](#) method.

Value	Meaning
SHI1005_FLAGS_DFS 0x00000001	The specified share is present in a <a href="#">DFS</a> tree structure.
SHI1005_FLAGS_DFS_ROOT 0x00000002	The specified share is present in a DFS tree structure.
CSC_MASK 0x00000030	Provides a mask for one of the four possible client-side caching (CSC) (section 2.2.2.5) states.
SHI1005_FLAGS_RESTRICT_EXCLUSIVE_OPENS 0x00000100	The specified share disallows exclusive file opens that deny reads to an open file.
SHI1005_FLAGS_FORCE_SHARED_DELETE 0x00000200	The specified share disallows clients from opening files on the share in an exclusive mode that prevents the file from being deleted until the client closes the file.
SHI1005_FLAGS_ALLOW_NAMESPACE_CACHING 0x00000400	Clients are allowed to cache the namespace of the specified share.
SHI1005_FLAGS_ACCESS_BASED_DIRECTORY_ENUM 0x00000800	The server filters directory entries based on the access permissions of the <a href="#">client.&lt;11&gt;</a>

Value	Meaning
SHI1005_FLAGS_FORCE_LEVELII_OPLOCK 0x00001000	The server does not issue exclusive caching rights on this share. <a href="#">&lt;12&gt;</a>
SHI1005_FLAGS_ENABLE_HASH 0x00002000	The share supports hash generation for branch cache retrieval of data. It is only valid if the server supports the branch cache capability and the branch cache component is installed. <a href="#">&lt;13&gt;</a>
SHI1005_FLAGS_ENABLE_CA 0x00004000	A highly available share. <a href="#">&lt;14&gt;</a>
SHI1005_FLAGS_ENCRYPT_DATA 0x00008000	A share on which remote file access is encrypted. <a href="#">&lt;15&gt;</a>
SHI1005_FLAGS_COMPRESS_DATA 0x00100000	A share on which remote file access is requested to be compressed.

## 2.2.4.30 SHARE\_INFO\_1006

Article02/14/2019

The SHARE\_INFO\_1006 structure specifies the maximum number of concurrent [connections](#) that the shared resource can accommodate. For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_1 \(section 2.2.4.26\)](#) structure (shi1006\_xxx denotes the same information as shi502\_xxx).

```
typedef struct _SHARE_INFO_1006 {
    DWORD shi1006_max_uses;
} SHARE_INFO_1006,
*PSHARE_INFO_1006,
*LPSHARE_INFO_1006;
```

## 2.2.4.31 SHARE\_INFO\_1501\_I

Article02/14/2019

The SHARE\_INFO\_1501\_I structure contains a security descriptor in self-relative format and a DWORD that contains its length.<16> For a description of the fields in this structure, see the description for the [SHARE\\_INFO\\_502\\_I \(section 2.2.4.26\)](#) structure (shi1501\_xxx denotes the same information as shi502\_xxx).

```
typedef struct _SHARE_INFO_1501_I {
    DWORD shi1501_reserved;
    [size_is(shi1501_reserved)] unsigned char* shi1501_security_descriptor;
} SHARE_INFO_1501_I,
*PSHARE_INFO_1501_I,
*LPSHARE_INFO_1501_I;
```

## 2.2.4.32 SHARE\_INFO\_0\_CONTAINER

Article02/14/2019

The SHARE\_INFO\_0\_CONTAINER structure contains a value that indicates the number of entries that the [NetrShareEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SHARE_INFO_0_CONTAINER {  
    DWORD EntriesRead;  
    [size_is(EntriesRead)] LPSHARE_INFO_0 Buffer;  
} SHARE_INFO_0_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SHARE\\_INFO\\_0](#) entries returned by the method.

## 2.2.4.33 SHARE\_INFO\_1\_CONTAINER

Article06/24/2021

The SHARE\_INFO\_1\_CONTAINER structure contains a value that indicates the number of entries that the [NetrShareEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SHARE_INFO_1_CONTAINER {  
    DWORD EntriesRead;  
    [size_is(EntriesRead)] LPSHARE_INFO_1 Buffer;  
} SHARE_INFO_1_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SHARE\\_INFO\\_1](#) entries returned by the method.

## 2.2.4.34 SHARE\_INFO\_2\_CONTAINER

Article02/14/2019

The SHARE\_INFO\_2\_CONTAINER structure contains a value that indicates the number of entries that the [NetrShareEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SHARE_INFO_2_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_2 Buffer;
} SHARE_INFO_2_CONTAINER,
*PSHARE_INFO_2_CONTAINER,
*LPSHARE_INFO_2_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SHARE\\_INFO\\_2](#) entries returned by the method.

## 2.2.4.35 SHARE\_INFO\_501\_CONTAINER

Article02/14/2019

The SHARE\_INFO\_501\_CONTAINER structure contains a value that indicates the number of entries that the [NetrShareEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SHARE_INFO_501_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_501 Buffer;
} SHARE_INFO_501_CONTAINER,
*PSHARE_INFO_501_CONTAINER,
*LPSHARE_INFO_501_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SHARE\\_INFO\\_501](#) entries returned by the method.

## 2.2.4.36 SHARE\_INFO\_502\_CONTAINER

Article02/14/2019

The SHARE\_INFO\_502\_CONTAINER structure contains a value that indicates the number of entries that the [NetrShareEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SHARE_INFO_502_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_502_I Buffer;
} SHARE_INFO_502_CONTAINER,
*PSHARE_INFO_502_CONTAINER,
*LPSHARE_INFO_502_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SHARE\\_INFO\\_502\\_I](#) entries returned by the method.

## 2.2.4.37 SHARE\_INFO\_503\_CONTAINER

Article02/14/2019

The SHARE\_INFO\_503\_CONTAINER structure contains a value that indicates the number of entries the [NetrShareEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SHARE_INFO_503_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_503_I Buffer;
} SHARE_INFO_503_CONTAINER,
*PSHARE_INFO_503_CONTAINER,
*LPSHARE_INFO_503_CONTAINER;
```

**EntriesRead:** The number of entries returned by the method.

**Buffer:** A pointer to the [SHARE\\_INFO\\_503\\_I](#) entries returned by the method.

## 2.2.4.38 SHARE\_ENUM\_STRUCT

Article10/30/2020

The SHARE\_ENUM\_STRUCT structure specifies the information level that the [client](#) requests in the [NetrShareEnum](#) method and encapsulates the [SHARE\\_ENUM\\_UNION](#) union that receives the entries enumerated by the [server](#).

```
typedef struct _SHARE_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] SHARE_ENUM_UNION ShareInfo;
} SHARE_ENUM_STRUCT,
*PSHARE_ENUM_STRUCT,
*LPSHARE_ENUM_STRUCT;
```

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
0	SHARE_INFO_0_CONTAINER
1	SHARE_INFO_1_CONTAINER
2	SHARE_INFO_2_CONTAINER
501	SHARE_INFO_501_CONTAINER
502	SHARE_INFO_502_CONTAINER
503	SHARE_INFO_503_CONTAINER

**ShareInfo:** Contains a share information container whose type is specified by the *Level* parameter as the preceding table shows. The enumerated share entries are returned in this member.

## 2.2.4.39 STAT\_SERVER\_0

Article 10/04/2021

The STAT\_SERVER\_0 structure contains statistical information about the [server](#).

```
typedef struct _STAT_SERVER_0 {
    DWORD sts0_start;
    DWORD sts0_fopens;
    DWORD sts0_devopens;
    DWORD sts0_jobsqueued;
    DWORD sts0_sopens;
    DWORD sts0_stimedout;
    DWORD sts0_serrorout;
    DWORD sts0_pwerrors;
    DWORD sts0_permerrors;
    DWORD sts0_syserrors;
    DWORD sts0_bytessent_low;
    DWORD sts0_bytessent_high;
    DWORD sts0_bytesrcvd_low;
    DWORD sts0_bytesrcvd_high;
    DWORD sts0_avresponse;
    DWORD sts0_reqbufneed;
    DWORD sts0_bigbufneed;
} STAT_SERVER_0,
*PSTAT_SERVER_0,
*LPSTAT_SERVER_0;
```

**sts0\_start:** Specifies a DWORD value that indicates the time when statistics collection started (or when the statistics were last cleared). The value MUST be stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, Greenwich Mean Time (GMT). To calculate the length of time that statistics have been collected, subtract the value of this member from the present time.

**sts0\_fopens:** Specifies a DWORD value that indicates the number of Opens that have been created on a server. This MUST include the number of times [named pipes](#) are opened.

**sts0\_devopens:** Specifies a DWORD value that indicates the number of times a server device has been opened. This field MUST be set to 0.

**sts0\_jobsqueued:** Specifies a DWORD value that indicates the number of server print jobs that are spooled.

**sts0\_sopens:** Specifies a DWORD value that indicates the number of sessions that have been established to a server.

**sts0\_stimedout:** Specifies a DWORD value that indicates the number of times a session is disconnected.

**sts0\_serrorout:** Specifies a DWORD value that indicates the number of times a session failed with an error. This field MUST be set to 0.

**sts0\_pwerrors:** Specifies a DWORD value that indicates the number of password violations that the server has detected.

**sts0\_permerrors:** Specifies a DWORD value that indicates the number of access permission errors that have occurred on the server.

**sts0\_syserrors:** Specifies a DWORD value that indicates the number of system errors that have occurred on the server. This field MUST be set to 0.

**sts0\_bytessent\_low:** Specifies the low-order DWORD of the number of server bytes sent on the network.

**sts0\_bytessent\_high:** Specifies the high-order DWORD of the number of server bytes sent on the network.

**sts0\_bytesrcvd\_low:** Specifies the low-order DWORD of the number of server bytes received from the network.

**sts0\_bytesrcvd\_high:** Specifies the high-order DWORD of the number of server bytes received from the network.

**sts0\_avresponse:** Specifies a DWORD value that indicates the average server response time, in milliseconds. This field MUST be set to 0.

**sts0\_reqbufneed:** Specifies a DWORD value that indicates the number of times the server required a request buffer but failed to allocate one. This field MUST be set to 0.

**sts0\_bigbufneed:** Specifies a DWORD value that indicates the number of times the server required a large buffer but failed to allocate one. This field MUST be set to 0.

## 2.2.4.40 SERVER\_INFO\_100

Article10/04/2021

The SERVER\_INFO\_100 structure contains information about the specified [server](#), including the name and platform. It MUST be used only to query information about a server. For a description of the fields in this structure, see the description for the [SERVER\\_INFO\\_103](#) structure (sv100\_xxx denotes the same information as sv103\_xxx).

```
typedef struct _SERVER_INFO_100 {
    DWORD sv100_platform_id;
    [string] wchar_t* sv100_name;
} SERVER_INFO_100,
*PSERVER_INFO_100,
*LPSERVER_INFO_100;
```

## 2.2.4.41 SERVER\_INFO\_101

Article02/14/2019

The SERVER\_INFO\_101 structure contains information about the specified [server](#), including name, platform, type of server, and associated software. For a description about the fields in this structure, see the description for the [SERVER\\_INFO\\_103](#) structure (sv101\_xxx denotes the same information as sv103\_xxx).

```
typedef struct _SERVER_INFO_101 {
    DWORD sv101_platform_id;
    [string] wchar_t* sv101_name;
    DWORD sv101_version_major;
    DWORD sv101_version_minor;
    DWORD sv101_type;
    [string] wchar_t* sv101_comment;
} SERVER_INFO_101,
*PSERVER_INFO_101,
*LPSERVER_INFO_101;
```

## 2.2.4.42 SERVER\_INFO\_102

Article02/14/2019

The SERVER\_INFO\_102 structure contains information about the specified [server](#), including the name, platform, and type of server, attributes, and associated software. For information about the fields in this structure, see the description for the [SERVER\\_INFO\\_103](#) structure (sv102\_xxx denotes the same information as sv103\_xxx).

```
typedef struct _SERVER_INFO_102 {
    DWORD sv102_platform_id;
    [string] wchar_t* sv102_name;
    DWORD sv102_version_major;
    DWORD sv102_version_minor;
    DWORD sv102_type;
    [string] wchar_t* sv102_comment;
    DWORD sv102_users;
    long sv102_disc;
    int sv102_hidden;
    DWORD sv102_announce;
    DWORD sv102_anndelta;
    DWORD sv102_licenses;
    [string] wchar_t* sv102_userpath;
} SERVER_INFO_102,
*PSERVER_INFO_102,
*LPSERVER_INFO_102;
```

## 2.2.4.43 SERVER\_INFO\_103

Article 10/04/2021

The SERVER\_INFO\_103 structure contains information about CIFS and SMB Version 1.0 file servers, including the name, platform, type of server, attributes, associated software, and capabilities.

```
typedef struct _SERVER_INFO_103 {
    DWORD sv103_platform_id;
    [string] wchar_t* sv103_name;
    DWORD sv103_version_major;
    DWORD sv103_version_minor;
    DWORD sv103_type;
    [string] wchar_t* sv103_comment;
    DWORD sv103_users;
    LONG sv103_disc;
    BOOL sv103_hidden;
    DWORD sv103_announce;
    DWORD sv103_anndelta;
    DWORD sv103_licenses;
    [string] wchar_t* sv103_userpath;
    DWORD sv103_capabilities;
} SERVER_INFO_103,
*PSERVER_INFO_103,
*LPSERVER_INFO_103;
```

**sv103\_platform\_id:** Specifies the information level to use for platform-specific information. This member can be one of the values that are listed in [PLATFORM IDs \(section 2.2.2.6\)](#). The server MUST ignore this field during a [NetrServerSetInfo](#) operation.

**sv103\_name:** A pointer to a null-terminated Unicode UTF-16 [Internet host name](#) or [NetBIOS host name](#) of a server.

The server MUST ignore this field during a NetrServerSetInfo operation.

**sv103\_version\_major:** Specifies the major release version number of the operating system. The server MUST ignore this field during a NetrServerSetInfo operation. The server MUST set this field to an implementation-specific major release version number that corresponds to the host operating system on a [NetrServerGetInfo](#) operation. <17>

**sv103\_version\_minor:** Specifies the minor release version number of the operating system. The server MUST ignore this field during a NetrServerSetInfo operation. The server MUST set this field to an implementation-specific minor release version number that corresponds to the host operating system on a NetrServerGetInfo operation. <18>

**sv103\_type:** Specifies the type of software the computer is running. This member MUST be a combination of one or more of the values that are listed in section [2.2.2.7](#). The server MUST ignore this field during a NetrServerSetInfo operation.

**sv103\_comment:** An optional pointer to a null-terminated Unicode UTF-16 string that specifies a comment that describes the [server](#).

**sv103\_users:** Specifies the number of users who can attempt to log on to the server. The range of values MUST be from 0x00000001 to 0xFFFFFFFF, inclusive. The server enforces a ceiling, based on the particular SKU that is running on the server, by taking a minimum of the specified value and the ceiling.

**sv103\_disc:** Specifies the automatic disconnect time, in minutes. A session MUST be disconnected if it is idle longer than the period of time that the **sv103\_disc** member specifies. If the value of **sv103\_disc** is SV\_NODISC (0xFFFFFFFF), an automatic disconnect MUST NOT be enabled. The range of values MUST be from 0x00000001 to 0xFFFFFFFF, inclusive.

**sv103\_hidden:** A Boolean that specifies whether the server is hidden or visible to other computers in the same network domain. It MUST be set to TRUE (1) to indicate that the server is hidden; or it MUST be set to FALSE (0) to indicate that the server is visible. The default value is FALSE (0).

**sv103\_announce:** Specifies the network announce rate, in seconds. This rate determines how often the server is announced to other computers on the network for discovery by using the CIFS Browser Protocol. For more information, see [\[MS-BRWS\]](#). The range of values MUST be from 1 to 65535, inclusive.

**sv103\_anndelta:** Specifies the delta value for the announce rate, in milliseconds. This value specifies how much the announce rate can vary from the period of time that is specified in the **sv103\_announce** member. The delta value enables the server to set randomly varied announce rates in the range **sv103\_announce** to **sv103\_announce+sv103\_anndelta**, inclusive, to prevent many servers from announcing at the same time. The range of values MUST be from 0 to 65535, inclusive.

**sv103\_licenses:** Unused. The server MUST ignore this field during a NetrServerSetInfo operation. The server MUST return 0 during a NetrServerGetInfo operation.

**sv103\_userpath:** A pointer to a null-terminated Unicode UTF-16 string that specifies the path to the user directories. Due to historical reasons, the default path is "c:\\". The client can set this field to any value. The server stores this string and returns it when queried. This field has no effect on the server.

**sv103\_capabilities:** Specifies the capabilities of the server. This MUST be a combination of zero or more of the following flags. The server MUST ignore this field during a NetrServerSetInfo operation. If the server does not support any of the following capabilities, it MUST set this field to 0x0000.

Value	Meaning
SRV_SUPPORT_HASH_GENERATION 0x0001	Hash generation for branch cache functionality is supported by the server.
SRV_HASH_GENERATION_ACTIVE 0x0002	The branch cache component is installed. <a href="#">&lt;19&gt;</a>

## 2.2.4.44 SERVER\_INFO\_502

Article02/14/2019

The SERVER\_INFO\_502 structure contains information about a specified [server](#). For a description of the fields in this structure, see the description for the [SERVER\\_INFO\\_599](#) structure (sv502\_xxx denotes the same information as sv599\_xxx).

```
typedef struct _SERVER_INFO_502 {
    DWORD sv502_sessopens;
    DWORD sv502_sessvcs;
    DWORD sv502_opensearch;
    DWORD sv502_sizreqbuf;
    DWORD sv502_initworkitems;
    DWORD sv502_maxworkitems;
    DWORD sv502_rawworkitems;
    DWORD sv502_irpstacksize;
    DWORD sv502_maxrawbuflen;
    DWORD sv502_sessusers;
    DWORD sv502_sessconns;
    DWORD sv502_maxpagedmemoryusage;
    DWORD sv502_maxnonpagedmemoryusage;
    int sv502_enablessoftcompat;
    int sv502_enableforcedlogoff;
    int sv502_timesource;
    int sv502_acceptdownlevelapis;
    int sv502_lmannounce;
} SERVER_INFO_502,
*PSERVER_INFO_502,
*LPSERVER_INFO_502;
```

## 2.2.4.45 SERVER\_INFO\_503

Article02/14/2019

The SERVER\_INFO\_503 structure contains information about a specified [server](#). For a description of the fields in this structure, see the description for the [SERVER\\_INFO\\_599](#) structure (sv503\_xxx denotes the same information as sv599\_xxx).

```
typedef struct _SERVER_INFO_503 {
    DWORD sv503_sessopens;
    DWORD sv503_sessvcs;
    DWORD sv503_opensearch;
    DWORD sv503_sizreqbuf;
    DWORD sv503_initworkitems;
    DWORD sv503_maxworkitems;
    DWORD sv503_rawworkitems;
    DWORD sv503_irpstacksize;
    DWORD sv503_maxrawbuflen;
    DWORD sv503_sessusers;
    DWORD sv503_sessconns;
    DWORD sv503_maxpagedmemoryusage;
    DWORD sv503_maxnonpagedmemoryusage;
    int sv503_enablessoftcompat;
    int sv503_enableforcedlogoff;
    int sv503_timesource;
    int sv503_acceptdownlevelapis;
    int sv503_lmannounce;
    [string] wchar_t* sv503_domain;
    DWORD sv503_maxcopyreadlen;
    DWORD sv503_maxcopywritelen;
    DWORD sv503_minkeepsearch;
    DWORD sv503_maxkeepsearch;
    DWORD sv503_minkeepcomplsearch;
    DWORD sv503_maxkeepcomplsearch;
    DWORD sv503_threadcountadd;
    DWORD sv503_numblockthreads;
    DWORD sv503_scavtimeout;
    DWORD sv503_minrcvqueue;
    DWORD sv503_minfreeworkitems;
    DWORD sv503_xactmemsize;
    DWORD sv503_threadpriority;
    DWORD sv503_maxmpxct;
    DWORD sv503_oplockbreakwait;
    DWORD sv503_oplockbreakresponsewait;
    int sv503_enableoplocks;
    int sv503_enableoplockforceclose;
    int sv503_enablefcbopens;
    int sv503_enableraw;
    int sv503_enablessharednetdrives;
    DWORD sv503_minfreeconnections;
    DWORD sv503_maxfreeconnections;
} SERVER_INFO_503,
```

```
*PSERVER_INFO_503,  
*LPSERVER_INFO_503;
```

## 2.2.4.46 SERVER\_INFO\_599

Article10/04/2021

The SERVER\_INFO\_599 structure contains information about a specified [server](#). The SERVER\_INFO\_599 fields involve implementation-specific details of CIFS and SMB Version 1.0 file servers. These fields can vary in how they apply to any given implementation. For more information, see section [3.1.4.18](#).

```
typedef struct _SERVER_INFO_599 {
    DWORD sv599_sessopens;
    DWORD sv599_sessvcs;
    DWORD sv599_opensearch;
    DWORD sv599_sizreqbuf;
    DWORD sv599_initworkitems;
    DWORD sv599_maxworkitems;
    DWORD sv599_rawworkitems;
    DWORD sv599_irpstacksize;
    DWORD sv599_maxrawbuflen;
    DWORD sv599_sessusers;
    DWORD sv599_sessconns;
    DWORD sv599_maxpagedmemoryusage;
    DWORD sv599_maxnonpagedmemoryusage;
    int sv599_enablessoftcompat;
    int sv599_enableforcedlogoff;
    int sv599_timesource;
    int sv599_acceptdownlevelapis;
    int sv599_lmannounce;
    [string] wchar_t* sv599_domain;
    DWORD sv599_maxcopyreadlen;
    DWORD sv599_maxcopywritelen;
    DWORD sv599_minkeepsearch;
    DWORD sv599_maxkeepsearch;
    DWORD sv599_minkeepcomplsearch;
    DWORD sv599_maxkeepcomplsearch;
    DWORD sv599_threadcountadd;
    DWORD sv599_numblockthreads;
    DWORD sv599_scavtimeout;
    DWORD sv599_minrcvqueue;
    DWORD sv599_minfreeworkitems;
    DWORD sv599_xactmemsize;
    DWORD sv599_threadpriority;
    DWORD sv599_maxmpxct;
    DWORD sv599_oplockbreakwait;
    DWORD sv599_oplockbreakresponsewait;
    int sv599_enableoplocks;
    int sv599_enableoplockforceclose;
    int sv599_enablefcbopens;
    int sv599_enableraw;
    int sv599_enablessharednetdrives;
    DWORD sv599_minfreeconnections;
```

```

        DWORD sv599_maxfreeconnections;
        DWORD sv599_initsesstable;
        DWORD sv599_initconntable;
        DWORD sv599_initfiletable;
        DWORD sv599_initsearchtable;
        DWORD sv599_alertschedule;
        DWORD sv599_errorthreshold;
        DWORD sv599_networkerrorthreshold;
        DWORD sv599_diskspacethreshold;
        DWORD sv599_reserved;
        DWORD sv599_maxlinkdelay;
        DWORD sv599_minlinkthroughput;
        DWORD sv599_linkinfovalidtime;
        DWORD sv599_scavqosinfoupdatetime;
        DWORD sv599_maxworkitemidletime;
    } SERVER_INFO_599,
    *PSERVER_INFO_599,
    *LPSERVER_INFO_599;

```

**sv599\_sessopens:** Specifies the number of files that can be open in one session. The range of values MUST be from 1 to 16384, inclusive.<20>

**sv599\_sessvcs:** Specifies the maximum number of sessions that are permitted per [client](#). This value MUST be set to one.

**sv599\_opensearch:** Specifies the number of search operations that can be carried out simultaneously. The range of values MUST be from 1 to 2,048, inclusive.

**sv599\_sizreqbuf:** Specifies the size, in bytes, of each server buffer. This field MUST be ignored by the server on receipt for set operations. The range of values MUST be 1,024 to 65,535, inclusive.<21>

**sv599\_initworkitems:** Specifies the initial number of receive buffers, or [work items](#), that the server uses. The range of values for get operations MUST be from 1 to 512, inclusive. This field MUST be ignored by the server on receipt for set operations.

**sv599\_maxworkitems:** Specifies the maximum number of receive buffers, or work items, that the server can allocate. If this limit is reached, the transport MUST initiate flow control. The range of values MUST be from 1 to 65,535, inclusive. The server enforces a ceiling based on the particular SKU that is running on the server by taking a minimum specified value and the ceiling.

**sv599\_rawworkitems:** Specifies the number of special work items the server uses for raw mode I/O. A larger value for this member can increase performance, but it requires more memory. The range of values for get operations MUST be from 1 to 512, inclusive. This field MUST be ignored by the server on receipt for set operations.

**sv599\_irpstacksize:** Specifies the number of stack locations that the server allocated in I/O request packets (IRPs). This field MUST be ignored by the server on receipt for set operations. The range of values MUST be 11 to 50, inclusive.<22>

**sv599\_maxrawbuflen:** The server MUST validate the value on receipt. This value MUST be set to 65,535. Due to historical reasons, the server does not store this value.

**sv599\_sessusers:** Specifies the maximum number of users who can be logged on to the server in a single [connection](#). The range of values MUST be from 1 to 2,048, inclusive.

**sv599\_sessconns:** Specifies the maximum number of tree connections that can be made on the server in a single session. The range of values MUST be from 1 to 2,048, inclusive.

**sv599\_maxpagedmemoryusage:** Specifies the maximum size of pageable memory, in bytes, that the server can allocate at any one time. The range of values MUST be from 0x00400000 to 0xFFFFFFFF, inclusive.<23>

**sv599\_maxnonpagedmemoryusage:** Specifies the maximum size of nonpaged memory in bytes that the server can allocate at any one time. The range of values MUST be from 0x00400000 to 0xFFFFFFFF, inclusive.<24>

**sv599\_enablessoftcompat:** A Boolean that specifies the SoftCompatibility capability of the server. This field MUST be set to TRUE (1) to enable the SoftCompatibility feature, or it MUST be set to FALSE (0) to disable the SoftCompatibility feature. The default value is TRUE (1). This setting affects the open mode when the client does not have read/write permission to the file it is accessing. If this feature is enabled, the server uses share access (parameter to CreateFile) equal to FILE\_SHARE\_READ and does not mark the open as compatibility mode open; otherwise, share access is set equal to 0, and the open is marked as compatibility mode open.

**sv599\_enableforcedlogoff:** A Boolean that specifies whether or not the server forces a client to disconnect, even if the client has open files, after the client's logon time has expired. This field MUST be set to TRUE (1) for the server to force a client to disconnect under those circumstances, or it MUST be set to FALSE (0) for the server not to force a client to disconnect under those circumstances. The default value is TRUE (1).

**sv599\_timesource:** A Boolean that specifies whether the server is a reliable time source.

**sv599\_acceptdownlevelapis:** A Boolean that specifies whether the server accepts method calls from previous-generation NTLM clients. This field MUST be set to TRUE (1) to enable the server to accept method calls from previous-generation NTLM clients, or it MUST be set to FALSE (0) to disable the server from accepting method calls from previous NTLM clients. The default value is TRUE (1). This field MUST be ignored by the server on receipt.

**sv599\_Imannounce:** A Boolean that specifies whether the server is visible to NTLM 2.x clients. The default value is FALSE (0). If this feature is enabled, the server announces its presence through LanMan or NetBIOS announcements.

**sv599\_domain:** A pointer to a Unicode UTF character string that specifies the name of the server's domain. This field cannot be modified by clients.

**sv599\_maxcopyreadlen:** The server MUST validate this value on receipt. The range of values MUST be from 0x00000000 to 0xFFFFFFFF, inclusive. Due to historical reasons, the server does not store this value.

**sv599\_maxcopywritelen:** The server MUST validate this value on receipt. The range of values MUST be from 0x00000000 to 0xFFFFFFFF, inclusive. Due to historical reasons, the server does not store this value.

**sv599\_minkeepsearch:** The server MUST validate this value on receipt. The range of values MUST be from 5 to 5,000, inclusive. Due to historical reasons, the server does not store this value.

**sv599\_maxkeepsearch:** Specifies the length of time, in seconds, that the server retains information about incomplete directory search operations. For more information about directory searches, see [\[MS-CIFS\]](#) sections [2.2.6.2](#) and [2.2.6.3](#). The range of values MUST be from 10 to 10,000, inclusive.

**sv599\_minkeepcomplsearch:** The server MUST validate this value on receipt. The range of values MUST be from 1 to 1,000, inclusive. Due to historical reasons, the server does not store this value.

**sv599\_maxkeepcomplsearch:** The server MUST validate this value on receipt. The range of values MUST be from 2 to 10,000, inclusive. Due to historical reasons, the server does not store this value.

**sv599\_threadcountadd:** Unused. This field MUST be ignored on receipt.

**sv599\_numblockthreads:** Unused. This field MUST be ignored on receipt.

**sv599\_scavtimeout:** Specifies the period of time, in seconds, that an implementation-specific timer on the server remains idle before waking up to service requests. This timer runs periodic maintenance tasks that monitor time-out requests, log errors, update server statistics, and update the connection [Quality of Service \(QoS\)](#) by querying the underlying transport. The range of values MUST be from 1 to 300, inclusive.

**sv599\_minrcvqueue:** Specifies the minimum number of free receive work items that the server requires before it begins to allocate more. The server keeps a pool of free work items for each worker queue. When a new request is posted to this queue, a work item

is picked from the pool to hold that request while it is being processed. The work item is returned to the pool after the processing is done. If the number of free work items (that is, work items that are not being used to process a request) for a queue falls below this setting, the server will request more work items to be allocated for the queue. The range of values MUST be from 0 to 10, inclusive.

**sv599\_minfreeworkitems:** Specifies the minimum number of available receive work items that the server requires to begin processing a [server message block](#). The range of values MUST be from 0 to 10, inclusive.

**sv599\_xactmemsize:** Specifies the size, in bytes, of the shared memory region that is used to process server methods. The range of values MUST be from 0x10000 (64 KB) to 0x1000000 (16 MB), inclusive. This field MUST be ignored by the server on receipt for set operations.

**sv599\_threadpriority:** Specifies the priority of all server threads in relation to the base priority of the process. The range of values MUST be from 0 to 15, inclusive. This field MUST be ignored by the server on receipt for set operations.

**sv599\_maxmpxct:** Specifies the maximum number of outstanding requests that any one client can send to the server. The range of values MUST be from 1 to 65,535, inclusive.

**sv599\_oplockbreakwait:** Specifies the period of time, in seconds, to wait before timing out an opportunistic lock break request. For more information about opportunistic locks, see [MS-CIFS] section [3.2.4.18](#). The range of values MUST be from 10 to 180, inclusive.

**sv599\_oplockbreakresponsewait:** Specifies the period of time, in seconds, that the server waits for a client to respond to an opportunistic lock break request from the server. For more information about opportunistic locks, see [MS-CIFS] section 3.2.4.18. The range of values MUST be from 10 to 180, inclusive.

**sv599\_enableoplocks:** A Boolean that specifies whether the server allows clients to use opportunistic locks on files. Opportunistic locks are a significant performance enhancement, but they have the potential to cause lost cached data on some networks, particularly wide-area networks. For more information about opportunistic locks, see [MS-CIFS] section 3.2.4.18. This field MUST be set to TRUE (1) to enable clients to use opportunistic locks on files, or it MUST be set to FALSE (0) to restrict clients from using opportunistic locks on files. The default value is TRUE (1).

**sv599\_enableoplockforceclose:** Unused. MUST be set to zero and ignored on receipt.

**sv599\_enablefcopens:** Specifies whether several MS-DOS File Control Blocks (FCBs) are placed in a single location accessible to the server. If enabled, this option can save

resources on the server. This field MUST be set to TRUE (1) to place multiple MS-DOS FCBs in a single location accessible to the server, and it MUST be set to FALSE (0) otherwise. The default value is TRUE (1).

**sv599\_enableraw:** Specifies whether the server processes raw SMBs. If enabled, this allows more data to transfer per transaction and improves performance. However, it is possible that processing raw SMBs can impede performance on certain networks. This field MUST be set to TRUE (1) to indicate that the server processes raw SMBs, and it MUST be set to FALSE (0) to indicate that the server does not process raw SMBs. The server MUST maintain the value of this member. The default value is TRUE (1).

**sv599\_enablessharednetdrives:** Specifies whether the server allows redirected server drives to be shared. The default value is FALSE (0).

**sv599\_minfreeconnections:** Specifies the minimum number of free connection blocks that are maintained per endpoint. The server MUST set these aside to handle bursts of requests by clients to connect to the server. The range of values MUST be from 2 to 1,024.<25>

**sv599\_maxfreeconnections:** Specifies the maximum number of free connection blocks that are maintained per endpoint. The server MUST set these aside to handle bursts of requests by clients to connect to the server. The range of values MUST be from 2 to 16,384.<26>

**sv599\_initsesstable:** Specifies the initial session table size for the server in terms of the number of records (session structures used by the server internally to represent active sessions). The range of values MUST be from 1 to 64, inclusive.

**sv599\_initconntable:** Specifies the initial connection table size for the server in terms of the number of records (connection structures used by the server internally to represent active connections). The range of values MUST be from 1 to 128, inclusive.

**sv599\_initfiletable:** Specifies the initial file table size for the server in terms of the number of records (file structures used by the server internally to represent current open resources). The range of values MUST be from 1 to 256, inclusive.

**sv599\_initsearchtable:** Specifies the initial search table size for the server in terms of the number of records (search structures used by the server internally to represent active searches). The range of values MUST be from 1 to 2,048, inclusive.

**sv599\_alertschedule:** Specifies the time, in minutes, between two invocations of an implementation-specific algorithm on the server. This algorithm monitors server errors and disk space limits, and it generates the implementation-specific failure events. The range of values MUST be from 1 to 65,535, inclusive.

**sv599\_errorthreshold:** Specifies the number of failed operations (non-network) that the server logs before raising an administrative alert. The particular operations whose failure causes the count of failed non-network operations to be incremented is implementation-dependent. The range of values MUST be from 1 to 65,535, inclusive.

**sv599\_networkerrorthreshold:** Specifies the minimum percentage of failed network operations that the server records before raising an administrative alert. An alert MUST be raised when (the number of failed network operations / the number of all attempted network operations) \* 100 is greater than or equal to this value. The range of values MUST be from 1 to 100, inclusive.

**sv599\_diskspacethreshold:** Specifies the percent of free disk at which to raise an administrative alert. The range of values MUST be from 0 to 99, inclusive.

**sv599\_reserved:** Reserved. This field MUST be set to zero.

**sv599\_maxlinkdelay:** Specifies the maximum link delay, in seconds, for the server. The server enables raw I/O [\[MS-SMB\]](#) for a connection only if oplocks are enabled for this connection and the link delay on the connection is less than or equal to this value. The range of values MUST be from 0x00000000 to 0x10000000, inclusive.

**sv599\_minlinkthroughput:** Specifies the minimum link throughput, in bytes/second, for the server. The server enables oplocks for a connection only if its current throughput is greater than or equal to this value. The range of values MUST be from 0x00000000 to 0xFFFFFFFF, inclusive.

**sv599\_linkinfovalidtime:** Specifies the time interval, in seconds, during which the server can use the computed link information before having to compute it again. The range of values MUST be from 0x00000000 to 0x10000000, inclusive.

**sv599\_scavqosinfoupdatetime:** Specifies the time interval for which an implementation-specific timer on the server has to update QoS information. This time interval allows the client to have the QoS information update done less frequently than the other tasks done by the timer. The range of values MUST be from 0x00000000 to 0x10000000, inclusive.

**sv599\_maxworkitemidletime:** Specifies the maximum work item idle time, in seconds. For historical reasons, the server only stores this value, and it has no effect on server operation. The range of values MUST be from 10 to 1,800, inclusive.

## 2.2.4.47 SERVER\_INFO\_1005

Article02/14/2019

The SERVER\_INFO\_1005 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1005 {
    [string] wchar_t* sv1005_comment;
} SERVER_INFO_1005,
*PSERVER_INFO_1005,
*LPSERVER_INFO_1005;
```

**sv1005\_comment:** This member is defined in the [sv103\\_comment](#) member in [SERVER\\_INFO\\_103](#) (section 2.2.4.43).

## 2.2.4.48 SERVER\_INFO\_1107

Article02/14/2019

The SERVER\_INFO\_1107 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1107 {  
    DWORD sv1107_users;  
} SERVER_INFO_1107,  
*PSERVER_INFO_1107,  
*LPSERVER_INFO_1107;
```

**sv1107\_users:** This member is defined in the [sv103\\_users](#) member in [SERVER\\_INFO\\_103](#) ([section 2.2.4.43](#)).

## 2.2.4.49 SERVER\_INFO\_1010

Article 10/04/2021

The SERVER\_INFO\_1010 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1010 {  
    long sv1010_disc;  
} SERVER_INFO_1010,  
*PSERVER_INFO_1010,  
*LPSERVER_INFO_1010;
```

**sv1010\_disc:** This member is defined in the [sv103\\_disc](#) member in [SERVER\\_INFO\\_103](#) ([section 2.2.4.43](#)).

## 2.2.4.50 SERVER\_INFO\_1016

Article02/14/2019

The SERVER\_INFO\_1016 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1016 {
    int sv1016_hidden;
} SERVER_INFO_1016,
*PSERVER_INFO_1016,
*LPSERVER_INFO_1016;
```

**sv1016\_hidden:** This member is defined in the [sv103\\_hidden](#) member in [SERVER\\_INFO\\_103](#) (section 2.2.4.43).

## 2.2.4.51 SERVER\_INFO\_1017

Article 02/14/2019

The SERVER\_INFO\_1017 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1017 {  
    DWORD sv1017_announce;  
} SERVER_INFO_1017,  
*PSERVER_INFO_1017,  
*LPSERVER_INFO_1017;
```

**sv1017\_announce:** This member is defined in the [sv103\\_announce](#) member in [SERVER\\_INFO\\_103](#) (section 2.2.4.43).

## 2.2.4.52 SERVER\_INFO\_1018

Article02/14/2019

The SERVER\_INFO\_1018 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1018 {  
    DWORD sv1018_anndelta;  
} SERVER_INFO_1018,  
*PSERVER_INFO_1018,  
*LPSERVER_INFO_1018;
```

**sv1018\_anndelta:** This member is defined in the [sv103\\_anndelta](#) member in [SERVER\\_INFO\\_103](#) (section 2.2.4.43).

## 2.2.4.53 SERVER\_INFO\_1501

Article02/14/2019

The SERVER\_INFO\_1501 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1501 {  
    DWORD sv1501_sessopens;  
} SERVER_INFO_1501,  
*PSERVER_INFO_1501,  
*LPSERVER_INFO_1501;
```

**sv1501\_sessopens:** This member is defined in the [sv599\\_sessopens](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.54 SERVER\_INFO\_1502

Article02/14/2019

The SERVER\_INFO\_1502 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1502 {  
    DWORD sv1502_sessvcs;  
} SERVER_INFO_1502,  
*PSERVER_INFO_1502,  
*LPSERVER_INFO_1502;
```

**sv1502\_sessvcs:** This member is defined in the [sv599\\_sessvcs](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.55 SERVER\_INFO\_1503

Article02/14/2019

The SERVER\_INFO\_1503 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1503 {  
    DWORD sv1503_opensearch;  
} SERVER_INFO_1503,  
*PSERVER_INFO_1503,  
*LPSERVER_INFO_1503;
```

**sv1503\_opensearch:** This member is defined in the [sv599\\_opensearch](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.56 SERVER\_INFO\_1506

Article02/14/2019

The SERVER\_INFO\_1506 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1506 {
    DWORD sv1506_maxworkitems;
} SERVER_INFO_1506,
*PSERVER_INFO_1506,
*LPSERVER_INFO_1506;
```

**sv1506\_maxworkitems:** This member is defined in the [sv599\\_maxworkitems](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.57 SERVER\_INFO\_1510

Article02/14/2019

The SERVER\_INFO\_1510 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1510 {  
    DWORD sv1510_sessusers;  
} SERVER_INFO_1510,  
*PSERVER_INFO_1510,  
*LPSERVER_INFO_1510;
```

**sv1510\_sessusers:** This member is defined in the [sv599\\_sessusers](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.58 SERVER\_INFO\_1511

Article04/06/2021

The SERVER\_INFO\_1511 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1511 {
    DWORD sv1511_sessconns;
} SERVER_INFO_1511,
*PSERVER_INFO_1511,
*LPSERVER_INFO_1511;
```

**sv1511\_sessconns:** This member is defined in the [sv599\\_sessconns](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.59 SERVER\_INFO\_1512

Article02/14/2019

The SERVER\_INFO\_1512 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1512 {  
    DWORD sv1512_maxnonpagedmemoryusage;  
} SERVER_INFO_1512,  
*PSERVER_INFO_1512,  
*LPSERVER_INFO_1512;
```

**sv1512\_maxnonpagedmemoryusage:** This member is defined in the **sv599\_maxnonpagedmemoryusage** member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.60 SERVER\_INFO\_1513

Article02/14/2019

The SERVER\_INFO\_1513 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1513 {  
    DWORD sv1513_maxpagedmemoryusage;  
} SERVER_INFO_1513,  
*PSERVER_INFO_1513,  
*LPSERVER_INFO_1513;
```

**sv1513\_maxpagedmemoryusage:** This member is defined in the [sv599\\_maxpagedmemoryusage](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.61 SERVER\_INFO\_1514

Article02/14/2019

The SERVER\_INFO\_1514 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1514 {  
    int sv1514_enablessoftcompat;  
} SERVER_INFO_1514,  
*PSERVER_INFO_1514,  
*LPSERVER_INFO_1514;
```

**sv1514\_enablessoftcompat:** This member is defined in the [sv599\\_enablessoftcompat](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.62 SERVER\_INFO\_1515

Article10/04/2021

The SERVER\_INFO\_1515 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1515 {  
    int sv1515_enableforcedlogoff;  
} SERVER_INFO_1515,  
*PSERVER_INFO_1515,  
*LPSERVER_INFO_1515;
```

**sv1515\_enableforcedlogoff:** This member is defined in the [sv599\\_enableforcedlogoff](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.63 SERVER\_INFO\_1516

Article02/14/2019

The SERVER\_INFO\_1516 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1516 {  
    int sv1516_timesource;  
} SERVER_INFO_1516,  
*PSERVER_INFO_1516,  
*LPSERVER_INFO_1516;
```

**sv1516\_timesource:** This member is defined in the [sv599\\_timesource](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.64 SERVER\_INFO\_1518

Article02/14/2019

The SERVER\_INFO\_1518 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1518 {
    int sv1518_lmannounce;
} SERVER_INFO_1518,
*PSERVER_INFO_1518,
*LPSERVER_INFO_1518;
```

**sv1518\_lmannounce:** This member is defined in the [sv599\\_lmannounce](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.65 SERVER\_INFO\_1523

Article02/14/2019

The SERVER\_INFO\_1523 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1523 {  
    DWORD sv1523_maxkeepsearch;  
} SERVER_INFO_1523,  
*PSERVER_INFO_1523,  
*LPSERVER_INFO_1523;
```

**sv1523\_maxkeepsearch:** This member is defined in the [sv599\\_maxkeepsearch](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.66 SERVER\_INFO\_1528

Article02/14/2019

The SERVER\_INFO\_1528 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1528 {  
    DWORD sv1528_scavtimeout;  
} SERVER_INFO_1528,  
*PSERVER_INFO_1528,  
*LPSERVER_INFO_1528;
```

**sv1528\_scavtimeout:** This member is defined in the [sv599\\_scavtimeout](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.67 SERVER\_INFO\_1529

Article02/14/2019

The SERVER\_INFO\_1529 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1529 {  
    DWORD sv1529_minrcvqueue;  
} SERVER_INFO_1529,  
*PSERVER_INFO_1529,  
*LPSERVER_INFO_1529;
```

**sv1529\_minrcvqueue:** This member is defined in the [sv599\\_minrcvqueue](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.68 SERVER\_INFO\_1530

Article 06/24/2021

The SERVER\_INFO\_1530 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1530 {  
    DWORD sv1530_minfreeworkitems;  
} SERVER_INFO_1530,  
*PSERVER_INFO_1530,  
*LPSERVER_INFO_1530;
```

**sv1530\_minfreeworkitems:** This member is defined in the [sv599\\_minfreeworkitems](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.69 SERVER\_INFO\_1533

Article02/14/2019

The SERVER\_INFO\_1533 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1533 {  
    DWORD sv1533_maxmpxct;  
} SERVER_INFO_1533,  
*PSERVER_INFO_1533,  
*LPSERVER_INFO_1533;
```

**sv1533\_maxmpxct:** This member is defined in the [sv599\\_maxmpxct](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.70 SERVER\_INFO\_1534

Article02/14/2019

The SERVER\_INFO\_1534 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1534 {  
    DWORD sv1534_oplockbreakwait;  
} SERVER_INFO_1534,  
*PSERVER_INFO_1534,  
*LPSERVER_INFO_1534;
```

**sv1534\_oplockbreakwait:** This member is defined in the [sv599\\_oplockbreakwait](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.71 SERVER\_INFO\_1535

Article02/14/2019

The SERVER\_INFO\_1535 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1535 {  
    DWORD sv1535_oplockbreakresponsewait;  
} SERVER_INFO_1535,  
*PSERVER_INFO_1535,  
*LPSERVER_INFO_1535;
```

**sv1535\_oplockbreakresponsewait:** This member is defined in the sv599\_oplockbreakresponsewait member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.72 SERVER\_INFO\_1536

Article 10/04/2021

The SERVER\_INFO\_1536 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1536 {
    int sv1536_enableoplocks;
} SERVER_INFO_1536,
*PSERVER_INFO_1536,
*LPSERVER_INFO_1536;
```

**sv1536\_enableoplocks:** This member is defined in the [sv599\\_enableoplocks](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.73 SERVER\_INFO\_1538

Article10/04/2021

The SERVER\_INFO\_1538 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1538 {  
    int sv1538_enablefcbopens;  
} SERVER_INFO_1538,  
*PSERVER_INFO_1538,  
*LPSERVER_INFO_1538;
```

**sv1538\_enablefcbopens:** This member is defined in the [sv599\\_enablefcbopens](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.74 SERVER\_INFO\_1539

Article02/14/2019

The SERVER\_INFO\_1539 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1539 {  
    int sv1539_enableraw;  
} SERVER_INFO_1539,  
*PSERVER_INFO_1539,  
*LPSERVER_INFO_1539;
```

**sv1539\_enableraw:** This member is defined in the [sv599\\_enableraw](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.75 SERVER\_INFO\_1540

Article02/14/2019

The SERVER\_INFO\_1540 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1540 {
    int sv1540_enables sharednetdrives;
} SERVER_INFO_1540,
*PSERVER_INFO_1540,
*LPSERVER_INFO_1540;
```

**sv1540\_enables sharednetdrives:** This member is defined in the [sv599\\_enables sharednetdrives](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.76 SERVER\_INFO\_1541

Article02/14/2019

The SERVER\_INFO\_1541 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1541 {
    int sv1541_minfreeconnections;
} SERVER_INFO_1541,
*PSERVER_INFO_1541,
*LPSERVER_INFO_1541;
```

**sv1541\_minfreeconnections:** This member is defined in the [sv599\\_minfreeconnections](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.77 SERVER\_INFO\_1542

Article06/24/2021

The SERVER\_INFO\_1542 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1542 {
    int sv1542_maxfreeconnections;
} SERVER_INFO_1542,
*PSERVER_INFO_1542,
*LPSERVER_INFO_1542;
```

**sv1542\_maxfreeconnections:** This member is defined in the [sv599\\_maxfreeconnections](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.78 SERVER\_INFO\_1543

Article 06/24/2021

The SERVER\_INFO\_1543 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1543 {  
    DWORD sv1543_initsesstable;  
} SERVER_INFO_1543,  
*PSERVER_INFO_1543,  
*LPSERVER_INFO_1543;
```

**sv1543\_initsesstable:** This member is defined in the [sv599\\_initsesstable](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.79 SERVER\_INFO\_1544

Article04/06/2021

The SERVER\_INFO\_1544 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1544 {  
    DWORD sv1544_initconntable;  
} SERVER_INFO_1544,  
*PSERVER_INFO_1544,  
*LPSERVER_INFO_1544;
```

**sv1544\_initconntable:** This member is defined in the [sv599\\_initconntable](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.80 SERVER\_INFO\_1545

Article02/14/2019

The SERVER\_INFO\_1545 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1545 {  
    DWORD sv1545_initfiletable;  
} SERVER_INFO_1545,  
*PSERVER_INFO_1545,  
*LPSERVER_INFO_1545;
```

**sv1545\_initfiletable:** This member is defined in the [sv599\\_initfiletable](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.81 SERVER\_INFO\_1546

Article02/14/2019

The SERVER\_INFO\_1546 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1546 {  
    DWORD sv1546_initsearchtable;  
} SERVER_INFO_1546,  
*PSERVER_INFO_1546,  
*LPSERVER_INFO_1546;
```

**sv1546\_initsearchtable:** This member is defined in the [sv599\\_initsearchtable](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.82 SERVER\_INFO\_1547

Article 10/04/2021

The SERVER\_INFO\_1547 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1547 {
    DWORD sv1547_alertschedule;
} SERVER_INFO_1547,
*PSERVER_INFO_1547,
*LPSERVER_INFO_1547;
```

**sv1547\_alertschedule:** This member is defined in the [sv599\\_alertschedule](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.83 SERVER\_INFO\_1548

Article02/14/2019

The SERVER\_INFO\_1548 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1548 {
    DWORD sv1548_errorthreshold;
} SERVER_INFO_1548,
*PSERVER_INFO_1548,
*LPSERVER_INFO_1548;
```

**sv1548\_errorthreshold:** This member is defined in the [sv599\\_errorthreshold](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.84 SERVER\_INFO\_1549

Article02/14/2019

The SERVER\_INFO\_1549 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1549 {
    DWORD sv1549_networkerrorthreshold;
} SERVER_INFO_1549,
*PSERVER_INFO_1549,
*LPSERVER_INFO_1549;
```

**sv1549\_networkerrorthreshold:** This member is defined in the **sv599\_networkerrorthreshold** member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.85 SERVER\_INFO\_1550

Article02/14/2019

The SERVER\_INFO\_1550 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1550 {  
    DWORD sv1550_diskspacethreshold;  
} SERVER_INFO_1550,  
*PSERVER_INFO_1550,  
*LPSERVER_INFO_1550;
```

**sv1550\_diskspacethreshold:** This member is defined in the [sv599\\_diskspacethreshold](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.86 SERVER\_INFO\_1552

Article02/14/2019

The SERVER\_INFO\_1552 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1552 {  
    DWORD sv1552_maxlinkdelay;  
} SERVER_INFO_1552,  
*PSERVER_INFO_1552,  
*LPSERVER_INFO_1552;
```

**sv1552\_maxlinkdelay:** This member is defined in the [sv599\\_maxlinkdelay](#) member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.87 SERVER\_INFO\_1553

Article 10/04/2021

The SERVER\_INFO\_1553 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1553 {  
    DWORD sv1553_minlinkthroughput;  
} SERVER_INFO_1553,  
*PSERVER_INFO_1553,  
*LPSERVER_INFO_1553;
```

**sv1553\_minlinkthroughput:** This member is defined in the [sv599\\_minlinkthroughput](#) member in [SERVER\\_INFO\\_599](#) (section 2.2.4.46).

## 2.2.4.88 SERVER\_INFO\_1554

Article02/14/2019

The SERVER\_INFO\_1554 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1554 {  
    DWORD sv1554_linkinfovalidtime;  
} SERVER_INFO_1554,  
*PSERVER_INFO_1554,  
*LPSERVER_INFO_1554;
```

**sv1554\_linkinfovalidtime:** This member is defined in the sv599\_linkinfovalidtime member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.89 SERVER\_INFO\_1555

Article02/14/2019

The SERVER\_INFO\_1555 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1555 {  
    DWORD sv1555_scavqosinfoupdatetime;  
} SERVER_INFO_1555,  
*PSERVER_INFO_1555,  
*LPSERVER_INFO_1555;
```

**sv1555\_scavqosinfoupdatetime:** This member is defined in the sv599\_scavqosinfoupdatetime member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.90 SERVER\_INFO\_1556

Article 02/14/2019

The SERVER\_INFO\_1556 structure contains information about a specified [server](#).

```
typedef struct _SERVER_INFO_1556 {
    DWORD sv1556_maxworkitemidletime;
} SERVER_INFO_1556,
*PSERVER_INFO_1556,
*LPSERVER_INFO_1556;
```

**sv1556\_maxworkitemidletime:** This member is defined in the **sv599\_maxworkitemidletime** member in [SERVER\\_INFO\\_599 \(section 2.2.4.46\)](#).

## 2.2.4.91 DISK\_INFO

Article02/14/2019

The DISK\_INFO structure contains information (the drive letter) about the disk device on the [server](#).

```
typedef struct _DISK_INFO {
    [string] WCHAR Disk[3];
} DISK_INFO,
*PDISK_INFO,
*LPDISK_INFO;
```

**Disk:** The drive identifier of the disk device. This MUST consist of two Unicode UTF-16 characters followed by the null-terminating character (for example, "A:\0"). The first character in this string MUST be a drive letter in the range "A" through "Z", inclusive. The second character MUST be the ":" character.

## 2.2.4.92 DISK\_ENUM\_CONTAINER

Article02/14/2019

The DISK\_ENUM\_CONTAINER structure contains a value that indicates the number of entries that the [NetrServerDiskEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _DISK_ENUM_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead), length_is(EntriesRead)]
    LPDISK_INFO Buffer;
} DISK_ENUM_CONTAINER;
```

**EntriesRead:** The number of entries that the method returns.

**Buffer:** A pointer to the [DISK\\_INFO](#) entries that the method returns.

## 2.2.4.93 SERVER\_TRANSPORT\_INFO\_0

Article02/14/2019

The SERVER\_TRANSPORT\_INFO\_0 structure contains information about the specified transport protocol, including the name, address, and location on the network. The definitions of fields in this structure are specified in section [2.2.4.96](#). Fields having names of the form svti0\_xxx MUST be defined as in the corresponding SERVER\_TRANSPORT\_INFO\_3 fields with names of the form svti3\_xxx.

```
typedef struct _SERVER_TRANSPORT_INFO_0 {
    DWORD svti0_numberofvcs;
    [string] wchar_t* svti0_transportname;
    [size_is(svti0_transportaddresslength)]
        unsigned char* svti0_transportaddress;
    DWORD svti0_transportaddresslength;
    [string] wchar_t* svti0_networkaddress;
} SERVER_TRANSPORT_INFO_0,
*PSERVER_TRANSPORT_INFO_0,
*LPSERVER_TRANSPORT_INFO_0;
```

## 2.2.4.94 SERVER\_TRANSPORT\_INFO\_1

Article02/14/2019

The SERVER\_TRANSPORT\_INFO\_1 structure contains information about the specified transport protocol, including the name, address, and location on the network. The definitions of fields in this structure are specified in section [2.2.4.96](#). Fields having names of the form svti1\_xxx MUST be defined as in the corresponding SERVER\_TRANSPORT\_INFO\_3 fields with names of the form svti3\_xxx.

```
typedef struct _SERVER_TRANSPORT_INFO_1 {
    DWORD svti1_numberofvcs;
    [string] wchar_t* svti1_transportname;
    [size_is(svti1_transportaddresslength)]
        unsigned char* svti1_transportaddress;
    DWORD svti1_transportaddresslength;
    [string] wchar_t* svti1_networkaddress;
    [string] wchar_t* svti1_domain;
} SERVER_TRANSPORT_INFO_1,
*PSERVER_TRANSPORT_INFO_1,
*LPSERVER_TRANSPORT_INFO_1;
```

## 2.2.4.95 SERVER\_TRANSPORT\_INFO\_2

Article02/14/2019

The SERVER\_TRANSPORT\_INFO\_2 structure contains information about the specified transport protocol, including the name and address. The definitions of fields in this structure are specified in section [2.2.4.96](#). Fields having names of the form svti2\_xxx MUST be defined as in the corresponding SERVER\_TRANSPORT\_INFO\_3 fields with names of the form svti3\_xxx.

```
typedef struct _SERVER_TRANSPORT_INFO_2 {
    DWORD svti2_numberofvcs;
    [string] wchar_t* svti2_transportname;
    [size_is(svti2_transportaddresslength)]
        unsigned char* svti2_transportaddress;
    DWORD svti2_transportaddresslength;
    [string] wchar_t* svti2_networkaddress;
    [string] wchar_t* svti2_domain;
    unsigned long svti2_flags;
} SERVER_TRANSPORT_INFO_2,
*PSERVER_TRANSPORT_INFO_2,
*LPSERVER_TRANSPORT_INFO_2;
```

## 2.2.4.96 SERVER\_TRANSPORT\_INFO\_3

Article 06/24/2021

The SERVER\_TRANSPORT\_INFO\_3 structure contains information about the specified transport protocol, including the name, address, and password (credentials).

```
typedef struct _SERVER_TRANSPORT_INFO_3 {
    DWORD svti3_numberofvcs;
    [string] wchar_t* svti3_transportname;
    [size_is(svti3_transportaddresslength)]
        unsigned char* svti3_transportaddress;
    DWORD svti3_transportaddresslength;
    [string] wchar_t* svti3_networkaddress;
    [string] wchar_t* svti3_domain;
    unsigned long svti3_flags;
    DWORD svti3_passwordlength;
    unsigned char svti3_password[256];
} SERVER_TRANSPORT_INFO_3,
*PSERVER_TRANSPORT_INFO_3,
*LPSERVER_TRANSPORT_INFO_3;
```

**svti3\_numberofvcs:** Specifies a **DWORD** value that indicates the number of [clients](#) that are connected to the [server](#) and that are using the transport protocol that is specified by the **svti3\_transportname** member.

**svti3\_transportname:** A pointer to a null-terminated Unicode string that contains the implementation-specific name of a device that implements support for the transport. This field is provided by the transport driver and can depend on the physical network adapter over which the transport runs. [<27>](#)

**svti3\_transportaddress:** A pointer to a variable that contains the transport address that the server is using on the transport device that is specified by the **svti3\_transportname** member. [<28>](#)

This member is usually the NetBIOS name that the server is using. In these instances, the name MUST be 16 characters long, and the last character MUST be a blank character (0x20).

**svti3\_transportaddresslength:** Specifies a **DWORD** value that contains the length, in bytes, of the **svti3\_transportaddress** member. [<29>](#)

**svti3\_networkaddress:** A pointer to a null-terminated character string that contains the address that the network adapter is using. The string is transport-specific. The server MUST ignore this field on receipt. [<30>](#)

**svti3\_domain:** A pointer to a null-terminated character string that contains the name of the domain to which the server announces its presence.

**svti3\_flags:** This member MUST be a combination of zero or more of the following values.

Value	Meaning
SVTI2_REMAP_PIPE NAMES 0x00000002	If this value is set for an <a href="#">endpoint</a> , client requests that arrive over the transport to open a <a href="#">named pipe</a> MUST be rerouted (remapped) to the local pipe name \$\$\backslash ServerName\backslash PipeName.
SVTI2_SCOPED_NAME 0x00000004	If this value is set for an endpoint, all shares attached to <a href="#">svti3_transportname</a> are <a href="#">scoped shares</a> .

**svti3\_passwordlength:** Specifies a **DWORD** value that indicates the number of valid bytes in the **svti3\_password** member.

**svti3\_password:** Specifies the credentials to use for the new transport address. If the **svti3\_passwordlength** member is zero, the credentials for the server MUST be used.

## 2.2.4.97

# SERVER\_XPORT\_INFO\_0\_CONTAINER

Article02/14/2019

The SERVER\_XPORT\_INFO\_0\_CONTAINER structure contains a value that indicates the number of entries that the [NetrServerTransportEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SERVER_XPORT_INFO_0_CONTAINER {  
    DWORD EntriesRead;  
    [size_is(EntriesRead)] LP SERVER_TRANSPORT_INFO_0 Buffer;  
} SERVER_XPORT_INFO_0_CONTAINER,  
*PSERVER_XPORT_INFO_0_CONTAINER;
```

**EntriesRead:** The number of entries that the method returns.

**Buffer:** A pointer to the [SERVER\\_TRANSPORT\\_INFO\\_0](#) entries that the method returns.

## 2.2.4.98

# SERVER\_XPORT\_INFO\_1\_CONTAINER

Article02/14/2019

The SERVER\_XPORT\_INFO\_1\_CONTAINER structure contains a value that indicates the number of entries that the [NetrServerTransportEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SERVER_XPORT_INFO_1_CONTAINER {  
    DWORD EntriesRead;  
    [size_is(EntriesRead)] LP SERVER_TRANSPORT_INFO_1 Buffer;  
} SERVER_XPORT_INFO_1_CONTAINER,  
*PSERVER_XPORT_INFO_1_CONTAINER;
```

**EntriesRead:** The number of entries that the method returns.

**Buffer:** A pointer to the [SERVER\\_TRANSPORT\\_INFO\\_1](#) entries that the method returns.

## 2.2.4.99

# SERVER\_XPORT\_INFO\_2\_CONTAINER

Article02/14/2019

The SERVER\_XPORT\_INFO\_2\_CONTAINER structure contains a value that indicates the number of entries that the [NetrServerTransportEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SERVER_XPORT_INFO_2_CONTAINER {  
    DWORD EntriesRead;  
    [size_is(EntriesRead)] LP SERVER_TRANSPORT_INFO_2 Buffer;  
} SERVER_XPORT_INFO_2_CONTAINER,  
*PSERVER_XPORT_INFO_2_CONTAINER;
```

**EntriesRead:** The number of entries that the method returns.

**Buffer:** A pointer to the [SERVER\\_TRANSPORT\\_INFO\\_2](#) entries that the method returns.

## 2.2.4.100

# SERVER\_XPORT\_INFO\_3\_CONTAINER

Article02/14/2019

The SERVER\_XPORT\_INFO\_3\_CONTAINER structure contains a value that indicates the number of entries that the [NetrServerTransportEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SERVER_XPORT_INFO_3_CONTAINER {  
    DWORD EntriesRead;  
    [size_is(EntriesRead)] LP SERVER_TRANSPORT_INFO_3 Buffer;  
} SERVER_XPORT_INFO_3_CONTAINER,  
*PSERVER_XPORT_INFO_3_CONTAINER;
```

**EntriesRead:** The number of entries that the method returns.

**Buffer:** A pointer to the [SERVER\\_TRANSPORT\\_INFO\\_3](#) entries that the method returns.

## 2.2.4.101 SERVER\_XPORT\_ENUM\_STRUCT

Article04/06/2021

The SERVER\_XPORT\_ENUM\_STRUCT structure specifies the information level that the client requests in the [NetrServerTransportEnum](#) method and encapsulates the SERVER\_XPORT\_ENUM\_UNION union that receives the entries that are enumerated by the server.

```
typedef struct _SERVER_XPORT_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] SERVER_XPORT_ENUM_UNION XportInfo;
} SERVER_XPORT_ENUM_STRUCT,
*PSERVER_XPORT_ENUM_STRUCT,
*LPSERVER_XPORT_ENUM_STRUCT;
```

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
0	SERVER_XPORT_INFO_0_CONTAINER
1	SERVER_XPORT_INFO_1_CONTAINER
2	SERVER_XPORT_INFO_2_CONTAINER
3	SERVER_XPORT_INFO_3_CONTAINER

**XportInfo:** Contains information about file server transports in the format that is determined by the *Level* parameter, as shown in the preceding table. This member receives the enumerated information.

## 2.2.4.102 SERVER\_ALIAS\_INFO\_0

Article10/04/2021

The SERVER\_ALIAS\_INFO\_0 structure contains the information about alias, including alias name and [server](#) target name.

```
typedef struct _SERVER_ALIAS_INFO_0 {
    [string] LMSTR srvai0_alias;
    [string] LMSTR srvai0_target;
    BOOLEAN srvai0_default;
    ULONG srvai0_reserved;
} SERVER_ALIAS_INFO_0,
*PSERVER_ALIAS_INFO_0,
*LPSERVER_ALIAS_INFO_0;
```

**srvai0\_alias:** An empty string or a pointer to a null-terminated Unicode UTF-16 string that specifies the name of a specified alias. It MUST be an empty string if **srvai0\_default** is nonzero and MUST be a non-empty string if **srvai0\_default** is 0.

**srvai0\_target:** A pointer to a null-terminated Unicode UTF-16 string. It specifies the server name that alias is attached to. The server MUST ignore this member when processing the [NetrServerAliasDel](#) method.

**srvai0\_default:** A [BOOLEAN](#) value. If it is set to TRUE, **srvai0\_target** MUST replace the default server name that is used to locate a [scoped share](#) in [NetrShareAdd](#)/[NetrShareDel](#)/[NetrShareSetInfo](#). If a scoped share cannot be found through a tuple of <share name, server name> due to a server name mismatch, the default server name is used in <share name, default server name> to continue scoped share searching. The server MUST ignore **srvai0\_default** when processing the [NetrServerAliasDel](#) method.

**srvai0\_reserved:** This field is not used. The server MUST ignore the value of this parameter on receipt.

## 2.2.4.103

# SERVER\_ALIAS\_INFO\_0\_CONTAINER

Article02/14/2019

The SERVER\_ALIAS\_INFO\_0\_CONTAINER structure contains a value that indicates the number of entries that the [NetrServerAliasEnum](#) method returns and a pointer to the buffer that contains the entries.

```
typedef struct _SERVER_ALIAS_INFO_0_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LP SERVER_ALIAS_INFO_0 Buffer;
} SERVER_ALIAS_INFO_0_CONTAINER;
```

**EntriesRead:** The number of entries that the method returns.

**Buffer:** A pointer to the [SERVER\\_ALIAS\\_INFO\\_0](#) entries that the method returns.

## 2.2.4.104 SERVER\_ALIAS\_ENUM\_STRUCT

Article10/30/2020

The SERVER\_ALIAS\_ENUM\_STRUCT structure specifies the information level that the client requests in the [NetrServerAliasEnum](#) method and encapsulates the SERVER\_ALIAS\_ENUM\_UNION union that receives the entries that are enumerated by the [server](#).

```
typedef struct _SERVER_ALIAS_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] union _SERVER_ALIAS_ENUM_UNION {
        [case(0)]
        SERVER_ALIAS_INFO_0_CONTAINER* Level0;
    } ServerAliasInfo;
} SERVER_ALIAS_ENUM_STRUCT,
*PSERVER_ALIAS_ENUM_STRUCT,
*LPSERVER_ALIAS_ENUM_STRUCT;
```

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
0	SERVER_ALIAS_INFO_0_CONTAINER

**ServerAliasInfo:** Contains information about server aliases in the format that is determined by the *Level* parameter, as shown in the preceding table. This member receives the enumerated information.

## 2.2.4.105 TIME\_OF\_DAY\_INFO

Article06/24/2021

The TIME\_OF\_DAY\_INFO structure contains information about the time of day from a remote [server](#).

```
typedef struct _TIME_OF_DAY_INFO {
    DWORD tod_elapsedt;
    DWORD tod_msecs;
    DWORD tod_hours;
    DWORD tod_mins;
    DWORD tod_secs;
    DWORD tod_hunds;
    long tod_timezone;
    DWORD tod_tinterval;
    DWORD tod_day;
    DWORD tod_month;
    DWORD tod_year;
    DWORD tod_weekday;
} TIME_OF_DAY_INFO,
*PTIME_OF_DAY_INFO,
*LPTIME_OF_DAY_INFO;
```

**tod\_elapsedt:** Specifies a **DWORD** value that contains the number of seconds since 00:00:00, January 1, 1970, GMT.

**tod\_msecs:** Specifies a **DWORD** value that contains the number of milliseconds from an arbitrary starting point (system reset).

**tod\_hours:** Specifies a **DWORD** value that contains the current hour. This value MUST be in the range 0 through 23, inclusive.

**tod\_mins:** Specifies a **DWORD** value that contains the current minute. This value MUST be in the range 0 through 59, inclusive.

**tod\_secs:** Specifies a **DWORD** value that contains the current second. This value MUST be in the range 0 through 59, inclusive.

**tod\_hunds:** Specifies a **DWORD** value that contains the current hundredth second (0.01 second). This value MUST be in the range 0 through 99, inclusive.

**tod\_timezone:** Specifies the time zone of the server. This value MUST be calculated, in minutes, from Greenwich Mean Time (GMT). For time zones that are west of Greenwich, the value MUST be positive; for time zones that are east of Greenwich, the value MUST be negative. A value of -1 MUST indicate that the time zone is undefined.

**tod\_tinterval**: Specifies a **DWORD** value that contains the time interval for each tick of the clock. Each integral integer MUST represent one ten-thousandth second (0.0001 second).

**tod\_day**: Specifies a **DWORD** value that contains the day of the month. This value MUST be in the range 1 through 31, inclusive.

**tod\_month**: Specifies a **DWORD** value that contains the month of the year. This value MUST be in the range 1 through 12, inclusive.

**tod\_year**: Specifies a **DWORD** value that contains the year.

**tod\_weekday**: Specifies a **DWORD** value that contains the day of the week. This value MUST be in the range 0 through 6, inclusive, where 0 is Sunday, 1 is Monday, and so on.

## 2.2.4.106 ADT\_SECURITY\_DESCRIPTOR

Article02/14/2019

The ADT\_SECURITY\_DESCRIPTOR structure contains a security descriptor in self-relative format and a value that includes the length of the buffer that contains the descriptor.

For more information, see [\[MS-DTYP\]](#) section 2.4.6.

```
typedef struct _ADT_SECURITY_DESCRIPTOR {
    DWORD Length;
    [size_is(Length)] unsigned char* Buffer;
} ADT_SECURITY_DESCRIPTOR,
*PADT_SECURITY_DESCRIPTOR;
```

**Length:** The length of the **Buffer** member.

**Buffer:** A buffer for the security descriptor in self-relative form. For more information, see [\[MS-DTYP\]](#) section 2.4.6.

## 2.2.4.107 NET\_DFS\_ENTRY\_ID

Article10/04/2021

The NET\_DFS\_ENTRY\_ID structure specifies a [DFS](#) local partition.

```
typedef struct _NET_DFS_ENTRY_ID {
    GUID Uid;
    [string] WCHAR* Prefix;
} NET_DFS_ENTRY_ID,
*LPNET_DFS_ENTRY_ID;
```

**Uid:** Specifies the unique identifier for the partition.

**Prefix:** A pointer to a null-terminated Unicode UTF-16 string that contains the path prefix for the partition.

## 2.2.4.108

# NET\_DFS\_ENTRY\_ID\_CONTAINER

Article02/14/2019

The NET\_DFS\_ENTRY\_ID\_CONTAINER structure contains a pointer to a buffer that contains [NET\\_DFS\\_ENTRY\\_ID](#) entries and a value that indicates the count of entries in the buffer.

```
typedef struct _NET_DFS_ENTRY_ID_CONTAINER {
    unsigned long Count;
    [size_is(Count)] LPNET_DFS_ENTRY_ID Buffer;
} NET_DFS_ENTRY_ID_CONTAINER,
*LPNET_DFS_ENTRY_ID_CONTAINER;
```

**Count:** The count of buffer array entries returned by the method.

**Buffer:** An array of NET\_DFS\_ENTRY\_ID entries returned by the method.

## 2.2.4.109 DFS\_SITENAME\_INFO

Article 10/30/2020

The DFS\_SITENAME\_INFO structure specifies a site name.

```
typedef struct _DFS_SITENAME_INFO {
    unsigned long SiteFlags;
    [string, unique] WCHAR* SiteName;
} DFS_SITENAME_INFO,
*PDFS_SITENAME_INFO,
*LPDFS_SITENAME_INFO;
```

**SiteFlags:** This member MUST be a combination of zero or more of the following values.

Value	Meaning
DFS_SITE_PRIMARY 0x00000001	The site name was returned by the DsrGetSiteName method, as specified in <a href="#">[MS-NRPC]</a> section <a href="#">3.5.4.3.6</a> .

**SiteName:** A pointer to a null-terminated Unicode UTF-16 string that specifies a unique site name.

## 2.2.4.110 DFS\_SITELIST\_INFO

Article02/14/2019

The DFS\_SITELIST\_INFO structure contains a value that indicates the count of entries and an array of DFS\_SITELIST\_INFO entries that the [NetrDfsManagerReportSiteInfo](#) method returns.

```
typedef struct _DFS_SITELIST_INFO {
    unsigned long cSites;
    [size_is(cSites)] DFS_SITENAME_INFO Site[];
} DFS_SITELIST_INFO,
*PDFS_SITELIST_INFO,
*LPDFS_SITELIST_INFO;
```

**cSites:** A count of site array entries returned by the method.

**Site:** An array of [DFS\\_SITENAME\\_INFO](#) entries that the method returns.

# 3 Protocol Details

Article 06/24/2021

The methods in this [RPC](#) interface all return 0x00000000 to indicate success and a nonzero, implementation-specific, error code to indicate failure. Unless otherwise specified, a server-side implementation of this protocol can choose any nonzero Win32 error value to signify an error condition, as specified in section [1.8](#). The [client](#) side of the Server Service Remote Protocol MUST NOT interpret returned error codes. The client side of the Server Service Remote Protocol MUST simply return error codes to the invoking application without taking any protocol action.

Note that the terms "client side" and "[server](#) side" refer to the initiating and receiving ends of the protocol respectively rather than to client or server versions of an operating system. These methods MUST all behave the same, regardless whether the server side of the protocol is running in a client or server version of an operating system.

## 3.1 Server Details

Article02/14/2019

The [server](#) responds to messages it receives from the [client](#).

## 3.1.1 Abstract Data Model

Article 06/24/2021

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The organization is provided to facilitate the explanation of how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behaviors are consistent with that described in this specification.

This data model requires elements to be synchronized with the Common Internet File System (CIFS) Protocol, the Server Message Block (SMB) Protocol, or the Server Message Block (SMB) Version 2 Protocol servers. This data model also requires that these protocols maintain these elements coherently with this data model at all times. An implementation that uses this data model has to observe atomicity requirements in order that all these protocols always share and maintain an identical view of the common data.

A [server](#) implementing this [RPC](#) interface contains several logical elements: an [SMB](#) file server, one or more network protocol transports, and a list of shared resources that the server is making available. There could also be virtual shares and services that provide SMB file server referrals for these virtual shares.<31>

One or more network protocol transports SHOULD be configured by a server implementing this RPC interface, to be associated with an SMB file server at its initialization.<32> A transport is a protocol that logically lies below the file server and provides reliable delivery of file server messages. If a transport is associated with a file server, it is said to be bound to or enabled for the server. The act of associating a transport with the file server is referred to as binding. The binding between a file server and a transport is represented by a "transport handle".

Transports can be dynamically bound (or enabled) and unbound (or disabled) from a file server. The server opens a transport handle when a transport is bound and closes it upon unbind. A transport MUST be bound to the file server for the server to receive messages through the transport. A transport has an implementation-specific name; transport names are unique on a per-computer basis.<33>

When a transport is bound to a file server, the server MUST perform the transport binding, as specified in [\[MS-SMB\]](#) section 2.1, for the requested transport.

The file server can make available multiple sets of resources (that is, files, printers, pipes, disks, and [mailslots](#)) for access by Common Internet File System (CIFS) [clients](#) over the network. Each set is referred to as a share and is identified by a unique network name.

Shares can be made dynamically available, and the act of making a share available is referred to as adding a share. Shares can also be made unavailable dynamically, which is referred to as removing a share. The server MUST keep a list of all active shares that are identified by a share identifier. If the share is marked as a [sticky share](#), the same information MUST be stored in persistent storage. The server MUST support two-phase deletion of shares.<34>

The SMB server assigns all objects (active sessions, [connections](#), opened resources, shares, and transports) unique identifiers. Identifiers are integer values that allow the server to uniquely identify the corresponding object. The server generates these identifiers when the corresponding object is created. The client obtains these identifiers in response to one of its requests (for example, an SMB client gets the session identifier in response to a Session Setup request) and then uses these identifiers in future requests to refer to the corresponding object. To support enumerating these objects, the server MUST store each of these objects in separate lists.

The server MUST keep track of several implementation-dependent statistics (as described by the [STAT\\_SERVER\\_0 \(section 2.2.4.39\)](#) structure) about the server performance that clients can query by calling the [NetrServerStatisticsGet](#) method.

If the server supports [DFS](#), as specified in [\[MS-DFSC\]](#), it MUST provide a software component called a DFS driver that processes all messages pertaining to DFS. These messages are specified in section [NetrDfsGetVersion](#) (Opnum 43) (section [3.1.4.35](#)) through section [NetrDfsManagerReportSiteInfo](#) (Opnum 52) (section [3.1.4.43](#)). The server MUST keep a list of the DFS shares and links and the associated information about the shares and links.

## 3.1.1.1 Global

Article 06/24/2021

The server MUST implement the following:

**AliasList:** A list of aliases in the server. Each element in the list is an **Alias** as defined in section [3.1.1.3](#).

**CifsInitialized:** A Boolean that indicates whether the CIFS or SMB server, as specified in [\[MS-CIFS\]](#), has completed its initialization. For more details, see section [3.1.6.14](#).

**NullSessionPipeList:** A list of [named pipe](#) names, without the "\pipe\" prefix, that an anonymous user is allowed to open. This list is queried by the Server Message Block (SMB) and SMB Version 2 protocols.

**DefaultServerName:** A null-terminated Unicode UTF-16 string that is used as a default server name to locate a [scoped share](#).

**FileList:** A list of **Opens**. Each element in the list is an **Open** as defined in section [3.1.1.6](#). Entries are inserted into the list as specified in section [3.1.6.4](#) and removed as specified in section [3.1.6.5](#).

**GlobalServerAnnounce:** A DWORD bitmask to indicate the services that are available on the server. It MUST be a combination of one or more of the values that are listed in section [2.2.2.7](#).

**PrinterShareCount:** A numeric value that indicates the number of printer shares on the server.

**ShareList:** A list of shares. Each element in the list is a **Share** as defined in section [3.1.1.7](#). Entries are inserted into the list as specified in section [3.1.4.7](#) and removed as specified in section [3.1.4.12](#) and section [3.1.4.15](#).

**SessionList:** A list of sessions. Each element in the list is a **Session** as defined in section [3.1.1.8](#). Entries are inserted into the list as specified in section [3.1.6.2](#) and removed as specified in section [3.1.6.3](#).

**Smb2Initialized:** A Boolean that indicates whether the SMB2 server, as specified in [\[MS-SMB2\]](#), has completed its initialization. For more details, see section [3.1.6.14](#).

**StatisticsStartTime:** A DWORD value indicating the time, in seconds, when the server statistics collection started.

**TransportList:** A list of transports. Each element in the list is a **Transport** ADM element as defined in section [3.1.1.2](#).

**TreeConnectList:** A list of tree connects. Each element in the list is a **TreeConnect** element defined in section [3.1.1.5](#). Entries are inserted into the list as specified in section [3.1.6.6](#) and removed as specified in section [3.1.6.7](#).

## 3.1.1.2 Per Transport

Article02/14/2019

This **Transport** element provides an abstraction of an underlying network transport protocol on which it listens for connections from [clients](#). The properties defined by this element MUST be persisted by the server.

The **Transport** element contains the following properties:

**Transport.Name:** An implementation-specific name used to refer to the transport.

**Transport.ServerName:** A null-terminated Unicode UTF-16 string that is used to identify the [server](#). It could be the server [NetBIOS host name](#), an IP address, [Domain Name System \(DNS\)](#), or a caller-supplied `svti*_transportaddress` provided by [NetrServerTransportAdd](#) or [NetrServerTransportAddEx](#).

The following are the acceptable forms of **Transport.ServerName**:

- NetBIOS name:
  - "EXAMPLE", see [\[RFC1001\]](#) and [\[RFC1002\]](#)
- IP address:
  - XXX.XXX.XXX.XXX
- DNS:
  - rs.internic.net, see [\[RFC1034\]](#) and [\[RFC1035\]](#)

**Transport.ConnectionCount:** The number of connections established using this transport.

**Transport.Flags:** A [DWORD](#) bitmask value containing zero or more of the values specified in section [2.2.4.96](#).

**Transport.Domain:** The name of the domain to which the server announces its presence.

### 3.1.1.3 Per Alias

Article02/14/2019

The [server](#) provides an alias for the existing server name through which the shared resource can be accessed.

**Alias.target:** The existing server name to which alias is attached. **Alias.target** must be a valid name for the server that matches a [Transport.ServerName](#) in the [TransportList](#).

**Alias.alias:** An alias name for **Alias.target** through which the shared resource can be accessed. **Alias.alias** MUST be unique in the [AliasList](#).

**Alias.default:** A Boolean value. If it is set to TRUE, [DefaultServerName](#) MUST be set to **Alias.target** if [DefaultServerName](#) is not NULL.

## 3.1.1.4 Server Properties Object (ServerConfiguration)

Article02/14/2019

The ServerConfiguration object maintains the [server](#) configuration information for CIFS and SMB Version 1.0 file servers. The properties defined by this object MUST be persisted by the server.

**ServerConfiguration.ServerInfo103:** All elements in this structure are as defined in section [2.2.4.43](#).

**ServerConfiguration.ServerInfo599:** All elements in this structure are as defined in section [2.2.4.46](#).

## 3.1.1.5 Per TreeConnect

Article02/14/2019

**GlobalTreeConnectId:** A local, unique 32-bit identifier generated to identify a TreeConnect.

## 3.1.1.6 Per Open

Article02/14/2019

**GlobalFileId:** A local, unique 32-bit identifier generated to identify an **Open**.

## 3.1.1.7 Per Share

Article 06/24/2021

The **Share** element maintains the following information for the shared resource (directory, named pipe, or printer):

**Share.ShareName:** The name for the shared resource on this [server](#).

**Share.ServerName:** The NetBIOS, fully qualified domain name (FQDN), or textual IPv4 or IPv6 address that the share is associated with. This value MUST be less than 256 characters in length. If the share is associated with the default computer name of the machine, the *ServerName* parameter MUST be set to "\*". For more information, see sections [1.3](#), [3.1.6.8](#), and [4.3](#).

**Share.IsPersistent:** A [BOOLEAN](#) value indicating whether the share is a [sticky share](#) (persistent).

**Share.IsMarkedForDeletion:** A [BOOLEAN](#) value indicating whether the share has been marked for deletion via the [NetrShareDelStart \(section 3.1.4.14\) RPC](#) method.

**Share.IsPrinterShare:** A [BOOLEAN](#) value indicating whether the share is a printer share.

**Share.LocalPath:** A path that describes the local resource that is being shared. This MUST be a store that either provides named pipe functionality, or that offers storage and/or retrieval of files. In the case of the latter, it can be a device that accepts a file and then processes it in some format, such as a printer.

**Share.FileSecurity:** An authorization policy, such as an access control list, that describes what actions users that connect to this share are allowed to perform on the shared resource. [\*<35>\*](#)

**Share.CscFlags:** The configured offline caching policy for this share. This value MUST be manual caching, automatic caching of files, automatic caching of files and programs, or no offline caching. For more information, see [\[MS-SMB2\]](#) section [2.2.10](#). For more information about offline caching, see [\[OFFLINE\]](#).

**Share.IsDfs:** A [BOOLEAN](#) that, if set, indicates that this share is configured for DFS. For more information, see [\[MSDFS\]](#).

**Share.DoAccessBasedDirectoryEnumeration:** A [BOOLEAN](#) that, if set, indicates that the results of directory enumerations on this share MUST be trimmed to include only the files and directories that the calling user has the right to access.

**Share.AllowNamespaceCaching:** A BOOLEAN that, if set, indicates that clients are allowed to cache directory enumeration results for better performance.

**Share.ForceSharedDelete:** A BOOLEAN that, if set, indicates that all opens on this share MUST include FILE\_SHARE\_DELETE in the sharing access.

**Share.RestrictExclusiveOpens:** A BOOLEAN that, if set, indicates that users who request read-only access to a file are not allowed to deny other readers.

**Share.Type:** The value indicates the type of share. It MUST be one of the values that are listed in section [2.2.2.4](#).

**Share.Remark:** A pointer to a null-terminated Unicode UTF-16 string that specifies an optional comment about the shared resource.

**Share.MaxUses:** The value indicates the maximum number of concurrent connections that the shared resource can accommodate.

**Share.CurrentUses:** The value indicates the number of current trees connected to the shared resource.

**Share.ForceLevel2Oplock:** A BOOLEAN that, if set, indicates that the server does not issue exclusive caching rights on this share.

**Share.HashEnabled:** A BOOLEAN that, if set, indicates that the share supports hash generation for branch cache retrieval of data.

## 3.1.1.8 Per Session

Article02/14/2019

**GlobalSessionId:** A locally unique 32-bit identifier generated to identify a **Session**.

# 3.1.1.9 Algorithm for Determining Path Type

Article 10/04/2021

The input for this algorithm is:

- **PathName:** A null-terminated UTF-16 string that specifies the path name to check in a case-insensitive manner.

The output for this algorithm is:

- **Type:** A path type value as specified in section [2.2.2.9](#) if the algorithm finds an appropriate path type; otherwise ERROR\_INVALID\_NAME (0x0000007B).

The pseudo code for the algorithm is shown in the following example.

```
// The following set of characters MUST be treated as invalid characters:  
<> " |  
If (PathName contains invalid character)  
Return ERROR_INVALID_NAME;  
    If (PathName begins with '\')  
        If (PathName begins with "\\\")  
            If (PathName begins with "\\.")  
                If (PathName begins with "\\.\\")  
                    If (Remaining part of the PathName contains '*' or '?')  
                        Return Type= ITYPE_PATH_ABSD_WC;  
                    Else  
                        Return Type= ITYPE_PATH_ABSD;  
                    EndIf  
                Else  
                    Return ERROR_INVALID_NAME;  
                EndIf  
            ElseIf ((PathName begins with "\\<computer-name>")  
// <computer-name> is any string other than ".")  
                If (PathName begins with "\\<computer-name>\")  
                    If (Remaining part of the PathName is not empty)  
                        If (Remaining part of the PathName contains '*' or '?')  
                            Return Type= ITYPE_UNC_WC_PATH;  
                        Else  
                            Return Type= ITYPE_UNC;  
                        EndIf  
                    EndIf  
                Else  
                    Return Type= ITYPE_UNC_COMPNAME;  
                EndIf  
            ElseIf ((PathName begins with "\\*")  
                If (PathName equals to "\\**")
```

```

        Return Type= ITYPE_UNC_WC;
    Else
        Return ERROR_INVALID_NAME;
    EndIf
EndIf

Else      // PathName begins with only single slash "\"
    If (PathName begins with "\DEV")
        If (PathName equals "\DEV\LPT<n>" or "\DEV\LPT<n>:")
            // <n> is any number, Examples: "\DEV\LPT1", "\DEV\LPT4:"
            Return Type= ITYPE_DEVICE_LPT;
        ElseIf (PathName equals "\DEV\COM<n>" or "\DEV\COM<n>:")
            // <n> is any number, Examples: "\DEV\COM1", "\DEV\COM4:"
            Return Type= ITYPE_DEVICE_COM;
        Else
            Return ERROR_INVALID_NAME;
        EndIf
    ElseIf (PathName contains '*' or '?')
        Return Type= ITYPE_PATH_ABSND_WC;
    Else
        Return Type= ITYPE_PATH_ABSND;
    EndIf
EndIf

ElseIf (PathName begins with [A-Z] followed by ':')// Examples: "C:", "f:"
    If (PathName equals "<drive>:") // <drive> is any letter
        Return ITYPE_DEVICE_DISK
    Else // (PathName = "<drive>:\...")
        If (Remaining part of the PathName after "<drive>:"contains '*' or '?')
            Return Type= ITYPE_PATH_ABSD_WC;
        Else
            Return Type= ITYPE_PATH_ABSD;
        EndIf
    EndIf

ElseIf (PathName equals "LPT<n>" or "LPT<n>:") //Examples: "LPT1", "lpt4:"
    Return Type= ITYPE_DEVICE_LPT;
ElseIf (PathName equals "COM<n>" or "COM<n>:") //Examples: "COM1", "com4:"
    Return Type= ITYPE_DEVICE_COM;
Else // Relative Paths
    If (PathName contains '*' or '?')
        Return Type= ITYPE_PATH_RELND_WC;
    Else
        Return Type= ITYPE_PATH_RELND;
    EndIf
EndIf

```

## 3.1.2 Timers

Article02/14/2019

None.

### 3.1.3 Initialization

Article 10/04/2021

The server MUST initialize **GlobalServerAnnounce** to SV\_TYPE\_SERVER. The server SHOULD combine any architecture-specific flags defined in section [2.2.2.7](#) to the **GlobalServerAnnounce** value using the bitwise OR operation.[\*\*<36>\*\*](#)

The server MUST initialize **PrinterShareCount** to 0.

The server MUST initialize **NullSessionPipeList** with implementation-specific defaults or with values from the persistent store.[\*\*<37>\*\*](#)

Guest account support is optional and can be disabled.

The server MUST set **CifsInitialized** to FALSE.

The server MUST set **Smb2Initialized** to FALSE.

The server MUST wait until **CifsInitialized** and **Smb2Initialized** are set to TRUE.[\*\*<38>\*\*](#)

The server MUST initialize **ServerConfiguration.ServerInfo103** as follows:

- **sv103\_name** MUST be set to the NetBIOS host name of the server.
- **sv103\_type** MUST be set to **GlobalServerAnnounce**.
- **sv103\_capabilities** MUST be set as follows.
  - If the server does not support SMB2 or does not support Content Information Retrieval requests as specified in [\[MS-SMB2\]](#) section [3.3.5.15.7](#), **sv103\_capabilities** MUST be set to 0.
  - If the server supports Content Information Retrieval requests but the local component that generates hashes locally is not installed, **sv103\_capabilities** MUST be set to SRV\_SUPPORT\_HASH\_GENERATION.
  - If the server supports Content Information Retrieval requests and the local component that generates hashes is installed, **sv103\_capabilities** MUST be set to (SRV\_SUPPORT\_HASH\_GENERATION | SRV\_HASH\_GENERATION\_ACTIVE).
- **sv103\_platform\_id**, **sv103\_version\_major**, **sv103\_version\_minor**, **sv103\_comment**, **sv103\_users**, **sv103\_disc**, **sv103\_hidden**, **sv103\_announce**, and **sv103\_anndelta** are initialized with implementation-specific defaults or with values from the persistent configuration store.[\*\*<39>\*\*](#)

The server MUST initialize `ServerConfiguration.ServerInfo599` with implementation-specific defaults or with values from the persistent store. <40>

The `server` MUST initialize `DefaultServerName` to NULL.

The server MUST initialize `TransportList` to an empty list.

The server MUST then read each `Transport` stored in the persistent store and construct a `SERVER_TRANSPORT_INFO_3` structure (specified in section 2.2.4.96) as follows:

- `svti3_numberofvcs` MUST be set to zero.
- `svti3_transportname` MUST be set to `Transport.Name`.
- `svti3_transportaddress` MUST be set to `Transport.ServerName`.
- `svti3_transportaddresslength` MUST be set to the length of `Transport.ServerName`.
- `svti3_networkaddress` MUST be set to NULL.
- `svti3_domain` MUST be set to `Transport.Domain`.
- `svti3_flags` MUST be set to `Transport.Flags`.

The server MUST then invoke the `NetrServerTransportAddEx` method specified in section 3.1.4.23, passing `SERVER_TRANSPORT_INFO_3` as the `Buffer` parameter and 3 as the `Level` parameter.

The server MUST initialize `TreeConnectList` to an empty list.

The server MUST initialize `FileList` to an empty list.

The server MUST initialize `SessionList` to an empty list.

The server MUST initialize `AliasList` to an empty list. The server MUST then add aliases stored in the persistent configuration store by invoking the `NetrServerAliasAdd` method specified in section 3.1.4.44 and passing the `InfoStruct` and `Level` parameters stored in the persistent configuration store.

The server MUST initialize `ShareList` to an empty list.

The server MUST then read each `Share` stored in the persistent store and construct a `SHARE_INFO_503_I` structure (specified in section 2.2.4.27) as follows:

- `share.shi503_netname` MUST be set to `Share.ShareName`.
- `share.shi503_type` MUST be set to `Share.Type`.

- `share.shi503_remark` MUST be set to `Share.Remark`.
- `share.shi503_permissions` MUST be set to 0.
- `share.shi503_max_uses` MUST be set to `Share.MaxUses`.
- `share.shi503_current_uses` MUST be set to 0.
- `share.shi503_path` MUST be set to `Share.LocalPath`.
- `share.shi503_passwd` MUST be set to NULL.
- `share.shi503_security_descriptor` MUST be set to `Share.FileSecurity`.
- `share.shi503_servername` MUST be set to `Share.ServerName`.

The server MUST then add shares by invoking the [NetrShareAdd](#) method specified in section 3.1.4.7 and passing the `SHARE_INFO_503_I` as *InfoStruct* and 503 as *Level* parameters.

The server MUST then construct a [SHARE\\_INFO\\_1005](#) structure (specified in section 2.2.4.29) as follows:

- `shi1005_flags` MUST be set to the result of bitwise AND of `CSC_MASK` and `Share.CscFlags`.
- `SHI1005_FLAGS_DFS` and `SHI1005_FLAGS_DFS_ROOT` bits in `shi1005_flags` MUST be set if `Share.IsDfs` is TRUE.
- `SHI1005_FLAGS_ACCESS_BASED_DIRECTORY_ENUM` bit in `shi1005_flags` MUST be set if `Share.DoAccessBasedDirectoryEnumeration` is TRUE.
- `SHI1005_FLAGS_ALLOW_NAMESPACE_CACHING` bit in `shi1005_flags` MUST be set if `Share.AllowNamespaceCaching` is TRUE.
- `SHI1005_FLAGS_FORCE_SHARED_DELETE` bit in `shi1005_flags` MUST be set if `Share.ForceSharedDelete` is TRUE.
- `SHI1005_FLAGS_RESTRICT_EXCLUSIVE_OPENS` bit in `shi1005_flags` MUST be set if `Share.RestrictExclusiveOpens` is TRUE.
- `SHI1005_FLAGS_ENABLE_HASH` bit in `shi1005_flags` MUST be set if `Share.HashEnabled` is TRUE.
- `SHI1005_FLAGS_FORCE_LEVELII_OPLOCK` bit in `shi1005_flags` MUST be set if `Share.ForceLevel2Oplock` is TRUE.

The server MUST then update shares by invoking the [NetrShareSetInfo](#) method specified in section 3.1.4.11 and passing the **SHARE\_INFO\_1005** as *InfoStruct* and 1005 as *Level* parameters.

The server MUST construct **SERVER\_INFO\_103** and **SERVER\_INFO\_599** structures from **ServerConfiguration.ServerInfo103** and **ServerConfiguration.ServerInfo599** respectively.

The server MUST update the SMB server configuration as specified in [MS-CIFS] section [3.3.4.22](#) by providing **SERVER\_INFO\_103** and **SERVER\_INFO\_599** structures as input parameters.

The server MUST enable the SMB server as specified in [MS-CIFS] section [3.3.4.18](#) and MUST set CifsEnabled to TRUE.

The server MUST enable the SMB2 server as specified in [MS-SMB2] section [3.3.4.22](#) and MUST set Smb2Enabled to TRUE.

The server MUST initialize **StatisticsStartTime** to the number of seconds that have elapsed since 00:00:00, January 1, 1970, Greenwich Mean Time (GMT).

# 3.1.4 Message Processing Events and Sequencing Rules

Article 10/04/2021

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 0
Opnum1NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 1
Opnum2NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 2
Opnum3NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 3
Opnum4NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 4
Opnum5NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 5
Opnum6NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 6
Opnum7NotUsedOnWire	Returns ERROR_NOT_SUPPORTED. Unused. Opnum: 7
NetrConnectionEnum	Lists all <a href="#">connections</a> made to a shared resource on the <a href="#">server</a> or all connections established from a particular computer. Opnum: 8
NetrFileEnum	Returns information about some or all open files on a server, depending on the parameters that are specified. Opnum: 9

Method	Description
<a href="#">NetrFileGetInfo</a>	Retrieves information about a particular opening of a server resource.  Opnum: 10
<a href="#">NetrFileClose</a>	Forces an open resource instance (for example, file, device, or <a href="#">named pipe</a> ) on the server to close.  Opnum: 11
<a href="#">NetrSessionEnum</a>	Provides information about sessions that are established on a server.  Opnum: 12
<a href="#">NetrSessionDel</a>	Ends a network session between a server and a <a href="#">client</a> .  Opnum: 13
<a href="#">NetrShareAdd</a>	Shares a server resource.  Opnum: 14
<a href="#">NetrShareEnum</a>	Retrieves information about each shared resource on a server.  Opnum: 15
<a href="#">NetrShareGetInfo</a>	Retrieves information about a particular shared resource on the server.  Opnum: 16
<a href="#">NetrShareSetInfo</a>	Sets the parameters of a shared resource.  Opnum: 17
<a href="#">NetrShareDel</a>	Deletes a share name from a server's list of shared resources, which disconnects all connections to the shared resource.  Opnum: 18
<a href="#">NetrShareDelSticky</a>	Deletes a <a href="#">sticky share</a> name from a server's list of shared resources, which disconnects all connections to the shared resource.  Opnum: 19
<a href="#">NetrShareCheck</a>	Checks whether a server is sharing a device.  Opnum: 20

Method	Description
<a href="#">NetrServerGetInfo</a>	Retrieves current configuration information for the specified server.  Opnum: 21
<a href="#">NetrServerSetInfo</a>	Sets a server's operating parameters.  Opnum: 22
<a href="#">NetrServerDiskEnum</a>	Retrieves a list of disk drives on a server.  Opnum: 23
<a href="#">NetrServerStatisticsGet</a>	Retrieves operating statistics for a service.  Opnum: 24
<a href="#">NetrServerTransportAdd</a>	Binds the server to the transport protocol.  Opnum: 25
<a href="#">NetrServerTransportEnum</a>	Supplies information about transport protocols that the server manages.  Opnum: 26
<a href="#">NetrServerTransportDel</a>	Unbinds (disconnects) the transport protocol from the server.  Opnum: 27
<a href="#">NetrRemoteTOD</a>	Returns the time of day information from a specified server.  Opnum: 28
<a href="#">Opnum29NotUsedOnWire</a>	Only used locally, never remotely.  Opnum: 29
<a href="#">NetprPathType</a>	Checks a path name to determine its type.  Opnum: 30
<a href="#">NetprPathCanonicalize</a>	Converts a path name to an implementation-specific canonical format.  Opnum: 31
<a href="#">NetprPathCompare</a>	Performs an implementation-specific comparison of two paths.  Opnum: 32

Method	Description
<a href="#">NetprNameValidate</a>	Performs implementation-specific checks to ensure that the specified name is a valid name for the specified type.  Opnum: 33
<a href="#">NetprNameCanonicalize</a>	Converts a name to an implementation-specific canonical format for the specified type.  Opnum: 34
<a href="#">NetprNameCompare</a>	Performs an implementation-specific comparison of two names of a specific name type.  Opnum: 35
<a href="#">NetrShareEnumSticky</a>	Retrieves information about each sticky shared resource on a server.  Opnum: 36
<a href="#">NetrShareDelStart</a>	Performs the initial phase of a two-phase share delete.  Opnum: 37
<a href="#">NetrShareDelCommit</a>	Performs the final phase of a two-phase share delete.  Opnum: 38
<a href="#">NetrpGetFileSecurity</a>	Returns a copy of the security descriptor protecting a file or directory.  Opnum: 39
<a href="#">NetrpSetFileSecurity</a>	Sets the security of a file or directory.  Opnum: 40
<a href="#">NetrServerTransportAddEx</a>	Binds the specified server to the transport protocol. This extended method allows the caller to specify information levels 1, 2, and 3 beyond what the NetrServerTransportAdd (section 3.1.4.22) method allows.  Opnum: 41
<a href="#">Opnum42NotUsedOnWire</a>	Only used locally, never remotely.  Opnum: 42

Method	Description
<a href="#">NetrDfsGetVersion</a>	Checks whether the server is a <a href="#">DFS</a> server, and if so, returns an implementation-specific DFS version.  Opnum: 43
<a href="#">NetrDfsCreateLocalPartition</a>	Marks a share as being a DFS share.  Opnum: 44
<a href="#">NetrDfsDeleteLocalPartition</a>	Deletes a DFS share (prefix) on the server.  Opnum: 45
<a href="#">NetrDfsSetLocalVolumeState</a>	Sets a local DFS share online or offline.  Opnum: 46
<a href="#">Opnum47NotUsedOnWire</a>	Unsupported and not defined. Unused.  Opnum: 47
<a href="#">NetrDfsCreateExitPoint</a>	Creates a <a href="#">DFS link</a> on the server.  Opnum: 48
<a href="#">NetrDfsDeleteExitPoint</a>	Deletes a DFS link on the server.  Opnum: 49
<a href="#">NetrDfsModifyPrefix</a>	Changes the path that corresponds to a DFS link on the server.  Opnum: 50
<a href="#">NetrDfsFixLocalVolume</a>	Adds knowledge of a new DFS share on the server.  Opnum: 51
<a href="#">NetrDfsManagerReportSiteInfo</a>	Gets Active Directory site information.  Opnum: 52
<a href="#">NetrServerTransportDelEx</a>	Unbinds (disconnects) the transport protocol from the server.  Opnum: 53
<a href="#">NetrServerAliasAdd</a>	Attaches an alias name to an existing server name.  Opnum: 54
<a href="#">NetrServerAliasEnum</a>	Retrieves alias information for a server.  Opnum: 55

Method	Description
<a href="#">NetrServerAliasDel</a>	Deletes an alias name from a server alias list. Opnum: 56
<a href="#">NetrShareDelEx</a>	Deletes a share name from a server's list of shared resources. Opnum: 57

An implementation MAY<sup><41></sup> choose to support the methods whose names begin with NetrDfs.

The methods MUST NOT throw an exception.

The server SHOULD enforce security measures to ensure that the caller has the required permissions to execute each method.<42>

## 3.1.4.1 NetrConnectionEnum (Opnum 8)

Article10/04/2021

The NetrConnectionEnum method lists all the **treeconnects** made to a shared resource on the **server** or all **treeconnects** established from a particular computer.

```
NET_API_STATUS NetrConnectionEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* Qualifier,
    [in, out] LPCONNECT_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE \(section 2.2.1.1\)](#) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Qualifier:** A pointer to a null-terminated UTF-16 string that specifies a share name or computer name for the [connections](#) of interest to the client.

**InfoStruct:** A pointer to a structure, in the format of a [CONNECT\\_ENUM\\_STRUCT \(section 2.2.4.5\)](#). The CONNECT\_ENUM\_STRUCT structure has a **Level** member that specifies the type of structure to return. The **Level** member MUST be one of the values specified in section 2.2.4.5.

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of the returned data. If the value that is specified is [MAX\\_PREFERRED\\_LENGTH \(section 2.2.2.2\)](#), the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that could have been enumerated if the buffer had been big enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle that is used to continue an existing TreeConnect search. The handle MUST be zero on the first call and left unchanged for subsequent calls. If ResumeHandle is NULL, a resume handle MUST NOT be stored. If this parameter is not NULL and the method returns **ERROR\_MORE\_DATA**, this parameter receives an implementation-specific nonzero value that can be passed in subsequent calls to this method to continue with the enumeration.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the [TreeConnectList](#).

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2.

In response to a NetrConnectionEnum request, the server MUST enumerate the tree connection entries in **TreeConnectList** based on the value of the *ResumeHandle* parameter. For each entry, the server MUST query **treeconnect** properties by invoking underlying server events as specified in [MS-CIFS] section 3.3.4.15 and [MS-SMB2] section 3.3.4.19, providing *TreeConnect.GlobalTreeConnectId* as the input parameter. When the server receives STATUS\_SUCCESS for a **treeConnect.GlobalTreeConnectId** from either a CIFS or SMB2 server, the server MUST consider the received CONNECTION\_INFO\_1 structure as valid, and it MUST continue to query all other **treeconnects** that are established on the server.

The server MUST filter the results of the queries based on the *Qualifier* input parameter:

The *Qualifier* parameter specifies a share name or computer name for **treeconnects** of interest to the client. If the *Qualifier* begins with "\\\", it is considered a computer name. Otherwise, it is considered a share name. Share names MUST NOT begin with "\\\".

If the *Qualifier* is the name of a share on the server, the server MUST return all **treeconnects** made to that share by returning only the entries where **treeconnect.coni1\_netname** matches with the *Qualifier*.

If the *Qualifier* is a computer name, the server MUST return all **treeconnects** made from the specified computer to the server by returning only the entries where **ServerName** matches with the *Qualifier*.

If the *Qualifier* parameter is a NULL (zero-length) string, or if the length of the *Qualifier* parameter (including the terminating null character) is greater than 1,024, the server MUST fail the call with ERROR\_INVALID\_PARAMETER.

The *Qualifier* parameter plays no role in determining the value of *ResumeHandle*. The server uses the *ResumeHandle* parameter to start the enumeration (as described in the processing rules that follow for the *ResumeHandle* parameter), and then applies the *Qualifier* parameter, if specified, to restrict the returned results to only those items that pass the qualifier test (as described previously in this topic for *Qualifier*) for share name or computer name.

The **InfoStruct** parameter has a **Level** member. The valid values of **Level** are 0 and 1. If the **Level** member is not equal to one of the valid values, the server MUST fail the call with ERROR\_INVALID\_LEVEL.

If the **Level** member is 0, the server MUST return the information about **treeconnects** by filling the **CONNECT\_INFO\_0\_CONTAINER** structure in the **ConnectInfo** field of the **InfoStruct** parameter as follows. The **CONNECT\_INFO\_0\_CONTAINER** structure contains an array of **CONNECTION\_INFO\_0** structures.

- **coni0\_id** MUST be set to **treeconnect.GlobalTreeConnectId**.

If the **Level** member is 1, the server MUST return the **treeconnects** by filling the **CONNECT\_INFO\_1\_CONTAINER** structure in the **ConnectInfo** field of the **InfoStruct** parameter. The **CONNECT\_INFO\_1\_CONTAINER** structure contains an array of **CONNECTION\_INFO\_1** structures.

The *PreferredMaximumLength* parameter specifies the maximum number of bytes that the server can return for the **ConnectInfo** buffer. If *PreferredMaximumLength* is insufficient to hold all the entries, the server MUST return the maximum number of entries that will fit in the **ConnectInfo** buffer and return **ERROR\_MORE\_DATA**. If this parameter is equal to **MAX\_PREFERRED\_LENGTH**, the server MUST return all the requested data.

If the server returns **NERR\_Success** or **ERROR\_MORE\_DATA**, it MUST set the *TotalEntries* parameter to equal the total number of entries passing the qualifier filter that could have been enumerated from the current resume position.

If *PreferredMaximumLength* is insufficient to hold all the entries and if the client has specified a *ResumeHandle* parameter, the server MUST set *ResumeHandle* to some implementation-specific value that allows the server to continue with this enumeration in the **TreeConnectList** on a subsequent call to this method with the same value for the *ResumeHandle* parameter.

The following rules specify processing of the *ResumeHandle* parameter:

- If the *ResumeHandle* parameter is either **NULL** or points to **0x00000000**, the enumeration MUST start from the beginning of the **TreeConnectList**.
- If the *ResumeHandle* parameter points to a nonzero value, the server MUST validate the *ResumeHandle*.
  - If the value of *ResumeHandle* is less than the size of the **TreeConnectList**, the server MUST continue enumeration based on the value of *ResumeHandle*. The value of *ResumeHandle* specifies the index value in the **TreeConnectList** after which enumeration is to begin.
  - If the value of *ResumeHandle* is greater than or equal to the size of the **TreeConnectList**, the server MUST return **NERR\_Success** and zero entries. fail the

call with `ERROR_INVALID_PARAMETER`.

- If the client specified a `ResumeHandle` and if the server returns `ERROR_MORE_DATA` (0x000000EA), the server MUST set `ResumeHandle` to the index value of the last enumerated `treeconnect` in the `TreeConnectList`.

Because the `ResumeHandle` specifies the index into the `TreeConnectList`, and the `TreeConnectList` can be modified between multiple requests, the results of a query spanning multiple requests using the `ResumeHandle` can be unreliable, resulting in either duplicate or missed active `treeconnects`.

The server SHOULD [43](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [44](#) fail the call.

## 3.1.4.2 NetrFileEnum (Opnum 9)

Article 06/24/2021

The NetrFileEnum method MUST return information about some or all open files on a [server](#), depending on the parameters specified, or return an error code.

```
NET_API_STATUS NetrFileEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* BasePath,
    [in, string, unique] WCHAR* UserName,
    [in, out] PFILE_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**BasePath:** A pointer to a null-terminated UTF-16 string that specifies a path component.

**UserName:** A pointer to a null-terminated UTF-16 string that specifies the name of a user.

**InfoStruct:** A pointer to a structure, in the format of a [FILE\\_ENUM\\_STRUCT](#). The FILE\_ENUM\_STRUCT structure has a **Level** field that specifies the type of structure to return. The **Level** member MUST be one of the values specified in section 2.2.4.10.

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of returned data. If the value that is specified is [MAX\\_PREFERRED\\_LENGTH](#), the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that could have been enumerated if the buffer had been big enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle that is used to continue an Open [connection](#) search. The handle MUST be zero on the first call and left unchanged for subsequent calls. If ResumeHandle is NULL, a resume handle MUST NOT be stored. If this parameter is not NULL and the method returns [ERROR\\_MORE\\_DATA](#), this parameter receives an implementation-specific nonzero value that can be passed in subsequent calls to this method to continue with the enumeration.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the list of the currently active connections.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x000000EA ERROR_MORE_DATA	The client request succeeded. More entries are available. Not all entries could be returned in the buffer size that is specified by <i>PreferredMaximumLength</i> .
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000084B NERR_BufTooSmall	The client request succeeded. More entries are available. The buffer size that is specified by <i>PreferredMaximumLength</i> was too small to fit even a single entry.

In response to a NetrFileEnum message, the server MUST enumerate **Open** entries in **FileList** based on the value of the *ResumeHandle* parameter. For each entry, the server MUST query open properties by invoking the underlying server events as specified in [MS-CIFS] section 3.3.4.16 and [MS-SMB2] section 3.3.4.20, providing *Open.GlobalFileId* as the input parameter. When the server receives STATUS\_SUCCESS for an **Open.GlobalFileId** from either a CIFS or SMB2 server, the server MUST consider the received FILE\_INFO\_3 structure as valid, and the server MUST continue to query all other open entries on the server. The server MUST then return the information about some or all valid open entries on a server, depending on the qualifier parameters that are specified.

The *BasePath* parameter specifies a qualifier for the returned information. If this parameter is not NULL, the server MUST return only those FILE\_INFO\_3 structures

received from CIFS and SMB2 servers, where the field `fi3_path_name` contains `BasePath` as the prefix. (A prefix is the path component up to a backslash.) If the `BasePath` parameter is not NULL and if the length of the `BasePath` string, including the terminating null character, is greater than 1,024, the server MUST fail the call with `ERROR_INVALID_PARAMETER`.

The `UserName` parameter MUST specify the name of a user. If this parameter is specified, the server MUST return only those `FILE_INFO_3` structures received from CIFS and SMB2 servers where the field `fi3_username` matches `UserName`. If the `UserName` parameter is not NULL and if the length of the `UserName` string, including the terminating null character, is greater than 1,024, the server MUST fail the call with `ERROR_INVALID_PARAMETER`.

The `BasePath` and `UserName` parameters have no role in determining the value of `ResumeHandle`. The server uses the `ResumeHandle` parameter to start the enumeration (as described in the rules that follow for processing the `ResumeHandle` parameter), and then applies these qualifier parameters, if specified, to restrict the returned results to only those items that pass the qualifier test (as described previously in this topic for `BasePath` and `UserName`) for returned information.

The `InfoStruct` parameter has a `Level` member. The valid values of `Level` are 2 and 3. If the `Level` member is not equal to one of the valid values, the server MUST fail the call with `ERROR_INVALID_LEVEL`.

The server MUST fill the return structures as follows.

If the `Level` member is 2, the server MUST return the information about `Opens` by filling the `FILE_INFO_2_CONTAINER` structure in the `FileInfo` field of the `InfoStruct` parameter as follows. The `FILE_INFO_2_CONTAINER` structure contains an array of `FILE_INFO_2` structures.

- `fi2_id` MUST be set to `open.fi3_id`.

If the `Level` member is 3, the server MUST return `Opens` directly by filling the `FILE_INFO_3_CONTAINER` structure in the `FileInfo` field of the `InfoStruct` parameter. The `FILE_INFO_3_CONTAINER` structure contains an array of `FILE_INFO_3` structures.

The `PreferredMaximumLength` parameter specifies the maximum number of bytes that the server can return for the `FileInfo` buffer.

If `PreferredMaximumLength` is insufficient to hold all the entries, the server MUST return the maximum number of entries that will fit in the `FileInfo` buffer and return `ERROR_MORE_DATA`. If this parameter is equal to `MAX_PREFERRED_LENGTH`, the server MUST return all the requested data.

If the server returns NERR\_Success or ERROR\_MORE\_DATA, it MUST set the *TotalEntries* parameter equal to the total number of entries passing the qualifier filter (*BasePath* or *UserName*) that could have been enumerated from the current resume position.

If the *PreferredMaximumLength* is insufficient to hold all the entries and if the client has specified a *ResumeHandle*, the server MUST set *ResumeHandle* to some implementation-specific value that allows the server to continue with this enumeration on a subsequent call to this method with the same value for *ResumeHandle*.

The following rules specify processing of the *ResumeHandle* parameter:

- If the *ResumeHandle* parameter is either NULL or points to 0x00000000, the enumeration MUST start from the beginning of the **FileList**.
- If the *ResumeHandle* parameter points to a nonzero value, the server MUST validate the *ResumeHandle*.
  - If the value of *ResumeHandle* is less than the size of the **FileList**, the server MUST continue enumeration based on the value of *ResumeHandle*. The value of *ResumeHandle* specifies the index into the **FileList** after which enumeration is to begin.
  - If the value of *ResumeHandle* is greater than or equal to the size of the **FileList**, the server MUST return NERR\_Success and zero entries.
- If the client specified a *ResumeHandle* and if the server returns ERROR\_MORE\_DATA (0x000000EA), the server MUST set the *ResumeHandle* to the index of the last enumerated file open in the **FileList**.

Because the *ResumeHandle* specifies the index into the **FileList**, and the **FileList** can be modified between multiple requests, the results of a query spanning multiple requests using the *ResumeHandle* can be unreliable, offering either duplicate or missed open files.

The server SHOULD [45](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [46](#) fail the call.

### 3.1.4.3 NetrFileGetInfo (Opnum 10)

Article 04/06/2021

The NetrFileGetInfo method MUST retrieve information about a particular open [server](#) resource or return an error code.

```
NET_API_STATUS NetrFileGetInfo(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD FileId,
    [in] DWORD Level,
    [out, switch_is(Level)] LPFILE_INFO InfoStruct
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**FileId:** Specifies the file identifier of the open resource to return information for. The value of this parameter MUST have been returned in a previous [NetrFileEnum](#) method call.

**NOTE:** The *FileId* parameter returned in a previous [NetrFileEnum](#) call is not guaranteed to be valid. Therefore, the [NetrFileGetInfo](#) method is not guaranteed to succeed based on the validity of the *FileId* parameter.

**Level:** Specifies the information level of the data. This parameter MUST have one of the following values.

Value	Meaning
2	<a href="#">FILE_INFO_2</a>
3	<a href="#">FILE_INFO_3</a>

**InfoStruct:** This parameter is of type [LPFILE\\_INFO](#), which is defined in section 2.2.3.3. Its contents are determined by the value of the **Level** member, as shown in the previous parameter table.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the file specified.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000084B NERR_BufTooSmall	The supplied buffer is too small.

In response to a `NetrFileGetInfo` message, the server MUST query open properties by invoking underlying server events as specified in [MS-CIFS] section 3.3.4.16 and [MS-SMB2] section 3.3.4.20, providing `FileId` as the input parameter. When the server receives a non-NUL `FILE_INFO_3` structure from either a CIFS or SMB2 server, the server MUST return information about a particular opening of a server resource (file, device, or [named pipe](#)). Otherwise, the server MUST fail the call with an `ERROR_FILE_NOT_FOUND` error code.

The `FileId` parameter specifies the file identifier of the open resource in `FileList` to return information for. The value of this parameter MUST have been returned in a previous `NetrFileEnum` message response by the server.

The `Level` parameter can be either 2 or 3. If the value of the `Level` parameter is anything else, the server MUST fail the call with `ERROR_INVALID_LEVEL`. The value of the `Level` parameter determines the format of the `InfoStruct` parameter.

The server MUST retrieve the `open` in `FILE_INFO_3` structure from CIFS and SMB2 servers and fill the return structures as follows.

If the value of the `Level` parameter is 2, the server MUST return information about the `open` whose file identifier is `FileId` by filling the `FILE_INFO_2` structure in the `FileInfo2` field of the `InfoStruct` parameter as follows:

- **fi2\_id** MUST be set to **open.fi3\_id**.

If the value of the *Level* parameter is 3, the server MUST return the **open** directly whose **fi3\_id** is equal to *FileId*.

The server SHOULD<47> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<48> fail the call.

## 3.1.4.4 NetrFileClose (Opnum 11)

Article 10/30/2020

The **server** receives the NetrFileClose method in an RPC\_REQUEST packet. In response, the server MUST force an open resource instance (for example, file, device, or [named pipe](#)) on the server to close. This message can be used when an error prevents closure by any other means.

```
NET_API_STATUS NetrFileClose(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD FileId
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) ([section 2.2.1.1](#)) pointer that identifies the server. The **client** MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**FileId:** Specifies the file identifier of the open file, device, or pipe to close.

**Note** The *FileId* parameter that is returned in a previous [NetrFileEnum](#) method call is not guaranteed to be valid. Therefore, the NetrFileClose method is not guaranteed to succeed based on the validity of the *FileId* parameter.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000090A NERR_FileIdNotFound	There is no open file with the specified identification number.

This message can be used when an error prevents closure by any other means.

The *FileId* parameter specifies the file identifier of the **Open** in **FileList** to close. The value of the *FileId* parameter MUST correspond to a *FileId* that is returned in a previous NetrFileEnum message response by the server. The server MUST look up **Open** in the **FileList** where *FileId* matches **Open.GlobalFileId**. If no match is found, the server MUST return NERR\_FileIdNotFound. If a match is found, the server MUST close the **Open** by invoking an underlying server event as specified in [MS-CIFS] section 3.3.4.13 or [MS-SMB2] section 3.3.4.17, providing *FileId* as the input parameter.

If either CIFS or SMB2 servers return STATUS\_SUCCESS, the server MUST return NERR\_Success. Otherwise, the server MUST fail the call with a NERR\_FileIdNotFound error code.

The server SHOULD <49> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD <50> fail the call.

## 3.1.4.5 NetrSessionEnum (Opnum 12)

Article 04/06/2021

The NetrSessionEnum method MUST return information about sessions that are established on a [server](#) or return an error code.

```
NET_API_STATUS NetrSessionEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* ClientName,
    [in, string, unique] WCHAR* UserName,
    [in, out] PSESSION_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**ClientName:** A pointer to a null-terminated UTF-16 string that specifies the name of the computer session for which information is to be returned. This string MUST be one of the following: a NULL (zero-length) string; or a string that MUST begin with \\.

**UserName:** A pointer to a null-terminated UTF-16 string that specifies the user name for which information is to be returned.

**InfoStruct:** A pointer to a structure, in the format of a [SESSION\\_ENUM\\_STRUCT](#). The SESSION\_ENUM\_STRUCT structure has a **Level** member that specifies the type of structure to return. The **Level** member MUST be one of the values specified in section 2.2.4.21.

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of the returned data. If the value that is specified is [MAX\\_PREFERRED\\_LENGTH](#), the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that could have been enumerated if the buffer had been big enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle that is used to continue an existing session search in [SessionList](#), as specified in section 3.1.1.1. The handle MUST be zero on the first call and remain unchanged for subsequent calls. If the *ResumeHandle* parameter is NULL, no resume handle MUST be stored. If this parameter

is not NULL and the method returns ERROR\_MORE\_DATA, this parameter receives an implementation-specific nonzero value that can be passed in subsequent calls to this method to continue with the enumeration.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the **SessionList**.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000000EA ERROR_MORE_DATA	The client request succeeded. More entries are available. Not all entries could be returned in the buffer size that is specified by <i>PreferredMaximumLength</i> .
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000908 NERR_ClientNameNotFound	A session does not exist with the computer name.
0x0000092F NERR_InvalidComputer	The computer name is not valid.
0x000008AD NERR_UserNotFound	The user name could not be found.

In response to the `NetrSessionEnum` message, the server MUST enumerate the **Session** entries in **SessionList** based on the value of the *ResumeHandle* parameter. For each entry, the server MUST query session properties by invoking the underlying server events as specified in [MS-CIFS] section 3.3.4.14 and [MS-SMB2] section 3.3.4.18, providing *Session.GlobalSessionId* as the input parameter. When the server receives a STATUS SUCCESS for a *Session.GlobalSessionId* from either a CIFS or SMB2 server, the server MUST consider the received `SESSION_INFO_502` structure as valid, and it MUST continue to query all other sessions that are established on the server. The server MUST then return information about some or all valid sessions that are established on the server, depending on the qualifier parameters that are specified.

The *ClientName* parameter specifies a qualifier for the returned information. If a *ClientName* is specified (that is, it is not a NULL (zero-length) string), the `sesi502_cname` field returned in the `SESSION_INFO_502` structure MUST match the *ClientName* for the session to be returned.

If a *ClientName* is specified, it MUST start with "\\"; otherwise, the server MUST fail the call with a `NERR_InvalidComputer` error code. If a *ClientName* is specified and it contains more than 1,024 characters, including the terminating null character, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The *UserName* parameter specifies a qualifier for the returned information. If a *UserName* is specified (that is, not a NULL (zero-length) string), the `sesi502_username` field returned in the `SESSION_INFO_502` structure MUST match the *UserName* parameter for the session to be returned. If a *UserName* parameter is specified and the length of the *UserName* string, including the terminating null character, is greater than 1,024 characters, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The server MUST return only those sessions that match all specified qualifiers. If no entries that match the qualifiers (*ClientName*/*UserName*) are found when a qualifier is specified, the server MUST fail the call with either an `NERR_UserNotFound` or `NERR_ClientNameNotFound` error code.

The *ClientName* and *UserName* parameters have no role in determining the value of *ResumeHandle*. The server uses the *ResumeHandle* parameter to start the enumeration (as described in the processing rules that follow for the *ResumeHandle* parameter), and then applies these qualifier parameters, if specified, to restrict the returned results to only those items that pass the qualifier test (as described previously in this topic for *ResumeHandle*).

The *InfoStruct* parameter has a **Level** member whose valid values are 0, 1, 2, 10, and 502. If the **Level** member is not equal to one of the valid values, the server MUST fail the call

with an `ERROR_INVALID_LEVEL` error code.

The server MUST fill the return structures as follows.

If the **Level** member is 0, the server MUST return the information about sessions by filling the `SESSION_INFO_0_CONTAINER` structure in the `SessionInfo` field of the `InfoStruct` parameter as follows. The `SESSION_INFO_0_CONTAINER` structure contains an array of `SESSION_INFO_0` structures.

- `sesi0_cname` MUST be set to `session.sesi502_cname`.

If the **Level** member is 1, the server MUST return the information about sessions by filling the `SESSION_INFO_1_CONTAINER` structure in the `SessionInfo` field of the `InfoStruct` parameter as in the following. The `SESSION_INFO_1_CONTAINER` structure contains an array of `SESSION_INFO_1` structures.

- `sesi1_cname` MUST be set to `session.sesi502_cname`.
- `sesi1_username` MUST be set to `session.sesi502_username`.
- `sesi1_numOpens` MUST be set to `session.sesi502_numOpens`.

If the **Level** member is 2, the server MUST return the information about sessions by filling the `SESSION_INFO_2_CONTAINER` structure in the `SessionInfo` field of the `InfoStruct` parameter as in the following. The `SESSION_INFO_2_CONTAINER` structure contains an array of `SESSION_INFO_2` structures.

- `sesi2_cname` MUST be set to `session.sesi502_cname`.
- `sesi2_username` MUST be set to `session.sesi502_username`.
- `sesi2_numOpens` MUST be set to `session.sesi502_numOpens`.
- `sesi2_idleTime` MUST be set to `session.sesi502_idletime`.
- `sesi2_time` MUST be set to `session.sesi502_time`.
- `sesi2_userFlags` MUST be set to `session.sesi502_user_flags`.
- `sesi2_cltype_name` MUST be set to `session.sesi502_cltype_name`.

If the **Level** member is 10, the server MUST return the information about sessions by filling the `SESSION_INFO_10_CONTAINER` structure in the `SessionInfo` field of the `InfoStruct` parameter as in the following. The `SESSION_INFO_10_CONTAINER` structure contains an array of `SESSION_INFO_10` structures.

- `sesi10_cname` MUST be set to `session.sesi502_cname`.

- `sesi10_username` MUST be set to `session.sesi502_username`.
- `sesi10_idle_time` MUST be set to `session.sesi502_idletime`.
- `sesi10_time` MUST be set to `session.sesi502_time`.

If the `Level` member is 502, the server MUST return the sessions in the `SESSION_INFO_502` structure by filling the `SESSION_INFO_502_CONTAINER` structure in the `SessionInfo` field of the `InfoStruct` parameter. The `SESSION_INFO_502_CONTAINER` structure contains an array of `SESSION_INFO_502` structures.

The `PreferredMaximumLength` parameter specifies the maximum number of bytes that the server can return for the `SessionInfo` buffer. If `PreferredMaximumLength` is insufficient to hold all the entries, the server MUST return the maximum number of entries that will fit in the `SessionInfo` buffer and return `ERROR_MORE_DATA`. If this parameter is equal to `MAX_PREFERRED_LENGTH`, the server MUST return all the requested data.

If the server returns `NERR_Success` or `ERROR_MORE_DATA`, it MUST set the `TotalEntries` parameter to equal the total number of entries that exceed the qualifier filter (`ClientName` or `UserName` as previously described) and that could have been enumerated from the current resume position.

If the `PreferredMaximumLength` is insufficient to hold all the entries and if the client has specified a `ResumeHandle`, the server MUST set `ResumeHandle` to some implementation-specific value that allows the server to continue with this enumeration on a subsequent call to this method with the same value for `ResumeHandle`.

The following rules specify processing of the `ResumeHandle` parameter:

- If the `ResumeHandle` parameter is either `NULL` or points to `0x00000000`, the enumeration MUST start from the beginning of the `SessionList`.
- If the `ResumeHandle` parameter points to a nonzero value, the server must validate the `ResumeHandle`.
  - If the value of `ResumeHandle` is less than the size of the `SessionList`, the server MUST continue enumeration based on the value of `ResumeHandle`. The value of `ResumeHandle` specifies the index into the `SessionList` after which enumeration is to begin.
  - If the value of `ResumeHandle` is greater than or equal to the size of the `SessionList`, the server MUST return `NERR_Success` and zero entries.
- If the client specified a `ResumeHandle` and the server returns `ERROR_MORE_DATA` (`0x000000EA`), the server MUST set `ResumeHandle` to the index value of the last

enumerated session in the **SessionList**.

Because the *ResumeHandle* specifies the index into the list and the list of active sessions can be modified between multiple requests, the results of a query spanning multiple requests using the *ResumeHandle* can be unreliable, offering either duplicate or inactive sessions.

The server SHOULD<51> enforce the security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<52> fail the call.

## 3.1.4.6 NetrSessionDel (Opnum 13)

Article 06/24/2021

The NetrSessionDel method MUST end one or more network sessions between a [server](#) and a [client](#).

```
NET_API_STATUS NetrSessionDel(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* ClientName,
    [in, string, unique] WCHAR* UserName
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) ([section 2.2.1.1](#)) pointer that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**ClientName:** A pointer to a null-terminated UTF-16 string that specifies the computer name of the client whose sessions are to be disconnected. This string MUST be one of the following: a NULL (zero-length) string; or a string that MUST begin with \\.

**UserName:** A pointer to a null-terminated UTF-16 string that specifies the user name whose sessions are to be terminated.

**Return Values:** This method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. This method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

Return value/code	Description
0x00000908	A session does not exist with the computer name.
NERR_ClientNameNotFound	

In response to a `NetrSessionDel` message, the server ends network sessions between the server and a workstation.

The server SHOULD[53](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD[54](#) fail the call.

The *ClientName* parameter specifies the computer name of the client to disconnect. If a *ClientName* is specified, it MUST start with "\\"; otherwise, the server MUST fail the call with an `NERR_ClientNameNotFound` error code. If a *ClientName* is specified and it contains more than 1,024 characters, including the terminating null character, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The *UserName* parameter specifies the name of the user whose session is to be terminated. If a *UserName* is specified and the length of the *UserName* string, including the terminating null character, is greater than 1,024, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

If both *ClientName* and *UserName* are unspecified (a NULL (zero-length) string), the server MUST fail the call with a `NERR_ClientNameNotFound` or an `ERROR_INVALID_PARAMETER` error code.

The server MUST enumerate all `Session` entries in `SessionList`. For each entry, the server MUST query session properties by invoking the underlying server events as specified in [MS-CIFS] section [3.3.4.14](#) and [MS-SMB2] section [3.3.4.18](#), providing

`Session.GlobalSessionId` as the input parameter. If the server receives a `STATUS_SUCCESS` for a `Session.GlobalSessionId` from either a CIFS or an SMB2 server, and the received `SESSION_INFO_502.sesi502_cname` matches the *ClientName* (if it is specified) and `SESSION_INFO_502.sesi502_username` matches the *UserName* (if it is specified), the server MUST close the session by invoking the underlying server event as specified in [MS-CIFS] section [3.3.4.8](#) or [MS-SMB2] section [3.3.4.12](#), providing `Session.GlobalSessionId` as input parameter. The server MUST continue to query all other sessions and close all the matching sessions.

If no matching session is found with the *ClientName* and *UserName*, the server MUST fail the call with error code `NERR_ClientNameNotFound`.

## 3.1.4.7 NetrShareAdd (Opnum 14)

Article 06/24/2021

The NetrShareAdd method shares a [server](#) resource.

```
NET_API_STATUS NetrShareAdd(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSHARE_INFO InfoStruct,
    [in, out, unique] DWORD* ParmErr
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. This parameter MUST be one of the following values.

Value	Meaning
2	The buffer is of type <a href="#">SHARE_INFO_2</a> .
502	The buffer is of type <a href="#">SHARE_INFO_502_I</a> .
503	The buffer is of type <a href="#">SHARE_INFO_503_I</a> .

**InfoStruct:** A pointer to the [SHARE\\_INFO](#) union. The contents of the *InfoStruct* parameter depend on the value of the *Level* parameter. The client MUST set the STYPE\_CLUSTER\_FS, STYPE\_CLUSTER\_SOFS, and STYPE\_CLUSTER\_DFS bits in the shi\*<sub>type</sub> field of the SHARE\_INFO union to zero; the server MUST ignore them on receipt.

**ParmErr:** A pointer to a value that receives the index of the first member of the share information structure that caused an ERROR\_INVALID\_PARAMETER error code, if it occurs.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
-------------------	-------------

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x0000007B ERROR_INVALID_NAME	The file name, directory name, or volume label syntax is incorrect.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid. For details, see the description that follows for the <i>ParmErr</i> parameter.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000846 NERR_DuplicateShare	The share name is already in use on this server.
0x00000844 NERR_UnknownDevDir	The device or directory does not exist.

In response to a `NetrShareAdd` message, the server MUST share a server resource or return an error code. A shared resource is a local resource on a server (for example, a disk directory, print device, or [named pipe](#)) that can be accessed by users and applications on the network.

The *Level* parameter determines the type of structure that the client has used to specify information about the new share. The value of the *Level* parameter MUST be 2, 502, or 503. If the *Level* parameter is not one of the valid values, the server MUST fail the call with an `ERROR_INVALID_LEVEL` error code.

If the *Level* parameter is 2, `InfoStruct` contains a `SHARE_INFO_2` structure.

If the *Level* parameter is 502, `InfoStruct` contains a `SHARE_INFO_502_I` structure.

If the *Level* parameter is 503, `InfoStruct` contains a `SHARE_INFO_503_I` structure.

The name of the share to be added is specified in the `shi*_netname` member of the `SHARE_INFO` structure. If the specified share name is an empty string, or is a nonempty string of length greater than 80 characters, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code. If the specified share name is "pipe" or "mailslot", the server MUST fail the call with an `ERROR_ACCESS_DENIED` error code.

If `Level` is 2 or 502, the server MUST look up the `Share` in `ShareList`, where `Share.ShareName` matches `shi*_netname` and `Share.ServerName` matches `"*"`.

If `Level` is 503, the server MUST look up the `Share` in `ShareList`, where `Share.ShareName` matches `shi503_netname` and `Share.ServerName` matches `shi503_servername`.

If a matching `Share` is found, the server MUST fail the call with `NERR_DuplicateShare`.

The server MUST validate all information that is provided in the `SHARE_INFO` (section 2.2.3.6) structure, and if any `SHARE_INFO` structure member is found to be invalid, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The server performs the following validation on the structure:

- `shi*_netname` must not be a NULL (zero-length) string, and its length must not be greater than 80 characters.
- If `Level=502` and a security descriptor is provided, it must be a valid security descriptor.
- If `shi*_netname` specifies an `IPC$` or the `ADMIN$` share, `shi*_path` must be `NULL`; otherwise, `shi*_path` must be a nonempty string that specifies a valid share path (must not have `".` and `..` appear as directory names).
- If `shi*_netname` specifies an NT path (begins with `"\\?\"`), `shi*_type` must not have a `STYPE_DISKTREE` flag.
- If `shi*_remark` is specified, its length must not be greater than 48.
- If `shi*_type` specifies a `STYPE_DISKTREE` flag and `shi*_netname` is not an `ADMIN$` share, `shi*_path` must specify an absolute directory path. If the server does not support shared net drivers (determined by the `SERVER_INFO` field `sv*_enablesharednetdrives`), the path must not be on a network drive.
- If a disk share is being added, the directory to be shared must exist and the caller must have access to it.

If the `ParmErr` parameter is not `NULL` and the server finds a member of the `SHARE_INFO` structure to be invalid, the server MUST set `ParmErr` to a value that denotes the index of

the member that was found to have an invalid value and fail the call with an ERROR\_INVALID\_PARAMETER (0x00000057) error code. The mapping between the values to set and the corresponding member is listed in section [2.2.2.11](#).

If the *ParmErr* parameter is NERR\_Success, the server MUST create a **Share** and insert it into **ShareList** with the following fields set:

- If the **STYPE\_TEMPORARY** field is set in *shi\*\_type*, **Share.IsPersistent** MUST be set to FALSE. Otherwise, **Share.IsPersistent** MUST be set to TRUE.
- **Share.IsMarkedForDeletion** MUST be set to FALSE.
- **Share.IsPrinterShare** MUST be set to TRUE if *shi\*\_type* specifies **STYPE\_PRINTQ** flag.
- **Share.ShareName** MUST be set to *shi\*\_netname*.
- **Share.ServerName** MUST be set to *shi503\_servername* if it is specified and if *Level* is equal to 503; otherwise it MUST be set to "\*".
- **Share.LocalPath** MUST be set to *shi\*\_path*.
- **Share.FileSecurity** MUST be set to *shi\*\_security\_descriptor* if it is specified and if *Level* is equal to 502 or 503; otherwise it MUST be set to NULL.
- **Share.CscFlags** MUST be set to 0.
- **Share.IsDfs** MUST be set to FALSE.
- **Share.DoAccessBasedDirectoryEnumeration** MUST be set to FALSE.
- **Share.AllowNamespaceCaching** MUST be set to FALSE.
- **Share.ForceSharedDelete** MUST be set to FALSE.
- **Share.RestrictExclusiveOpens** MUST be set to FALSE.
- **Share.Type** MUST be set to *shi\*\_type*.
- **Share.Remark** MUST be set to *shi\*\_remark*.
- **Share.MaxUses** MUST be set to 0xFFFF if *shi\*\_max\_uses* is not specified; otherwise it MUST be set to *shi\*\_max\_uses*.
- **Share.CurrentUses** MUST be set to 0.
- **Share.ForceLevel2Olock** MUST be set to FALSE.

If `shi*_type` specifies `STYPE_PRINTQ` flag, `PrinterShareCount` MUST be increased by 1, and the server MUST invoke the events as specified in section [3.1.6.9](#), providing `SV_TYPE_PRINTQ_SERVER` as the input parameter.

The server MUST construct a share in `SHARE_INFO_503_I` structure as the input parameter to register the share by invoking underlying server event as specified in [MS-CIFS] section [3.3.4.9](#) and [MS-SMB2] section [3.3.4.13](#), providing `share` as the input parameter. The fields in `share` MUST be set as follows:

- `share.shi503_netname` MUST be set to `Share.ShareName`.
- `share.shi503_type` MUST be set to `Share.Type`.
- `share.shi503_remark` MUST be set to `Share.Remark`.
- `share.shi503_permissions` MUST be set to 0.
- `share.shi503_max_uses` MUST be set to `Share.MaxUses`.
- `share.shi503_current_uses` MUST be set to 0.
- `share.shi503_path` MUST be set to `Share.LocalPath`.
- `share.shi503_passwd` MUST be set to NULL.
- `share.shi503_security_descriptor` MUST be set to `Share.FileSecurity`.
- `share.shi503_servername` MUST be set to `Share.ServerName`.

If either the CIFS or the SMB2 server returns an error:

- The server MUST remove the Share from `ShareList` and free the `share` object.
- The server MUST invoke the underlying server events as specified in [MS-CIFS] section [3.3.4.11](#) and [MS-SMB2] section [3.3.4.15](#), providing tuple `<Share.ServerName, Share.ShareName>` as input parameters.
- If the error returned by the CIFS or the SMB2 server is `STATUS_INVALID_PARAMETER`, then the server MUST fail the call with `ERROR_INVALID_DATA` (0x0000000D). Otherwise, the server MUST fail the call with `NERR_DuplicateShare`.

If `Share.IsPersistent` is TRUE, the server MUST persist the `Share` to a persistent configuration store. If a share with the same `ShareName` already exists in the store, the preexisting entry MUST be overwritten with this entry.

The server SHOULD<55> enforce the security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD<56> fail the call.

## 3.1.4.8 NetrShareEnum (Opnum 15)

Article 06/24/2021

The NetrShareEnum method retrieves information about each shared resource on a server.

```
NET_API_STATUS NetrShareEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, out] LPSHARE_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**InfoStruct:** A pointer to a structure, in the format of a [SHARE\\_ENUM\\_STRUCT](#) (section 2.2.4.38), as specified in section 2.2.4.38. The **SHARE\_ENUM\_STRUCT** structure has a **Level** member that specifies the type of structure to return in the **ShareInfo** member. The **Level** member MUST be one of the values specified in section 2.2.4.38.

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of the returned data. If the specified value is MAX\_PREFERRED\_LENGTH, the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that could have been enumerated if the buffer had been big enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle, which is used to continue an existing share search in **ShareList**. The handle MUST be zero on the first call and remain unchanged for subsequent calls. If the *ResumeHandle* parameter is NULL, no resume handle MUST be stored. If this parameter is not NULL and the method returns ERROR\_MORE\_DATA, this parameter receives a nonzero value that can be passed in subsequent calls to this method to continue with the enumeration in **ShareList**.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the **ShareList**.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code

value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x000000EA ERROR_MORE_DATA	The client request succeeded. More entries are available. Not all entries could be returned in the buffer size that is specified by <i>PreferredMaximumLength</i> .
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.

If *ServerName* does not match any **Transport.ServerName** in **TransportList** with the **SVTI2\_SCOPED\_NAME** bit set in **Transport.Flags**, the server MUST reset *ServerName* as **"\*"**.

The server MUST remove any preceding "\\" from the *ServerName* parameter and normalize the *ServerName* parameter as specified in section 3.1.6.8, passing in the updated *ServerName* parameter as the *ServerName*, and an empty string as the *ShareName*.

In response to a **NetrShareEnum** request, the server MUST enumerate the Share entries in **ShareList** based on the value of the **ResumeHandle** parameter and query share properties 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, providing the tuple *<normalized server name, Share.ShareName>* as the input parameter. When the server receives **STATUS\_SUCCESS** for a share, it MUST consider the received **SHARE\_INFO\_503\_I** and **SHARE\_INFO\_1005** structures as valid. The server MUST return information about each shared resource on a server.

The *InfoStruct* parameter has a **Level** member. The valid values of **Level** are 0, 1, 2, 501, 502, and 503. If the **Level** member is not equal to one of the valid values, the server MUST fail the call with an **ERROR\_INVALID\_LEVEL** error code.

The server MUST use the shares in valid **SHARE\_INFO\_503\_I** and **SHARE\_INFO\_1005** structures returned from either CIFS or SMB2 server and fill the return structures as follows. For each **share**, the server MUST discard the structures received from other file server except the value of **share.shi503\_current\_uses**.

If the **Level** member is 503, the server MUST return all shares in **SHARE\_INFO\_503\_I** structures. Otherwise, the server MUST return the **shares** in which

`share.shi503_servername` matches *ServerName*.

If the **Level** member is 0, the server MUST return the information about **share** resources by filling the **SHARE\_INFO\_0\_CONTAINER** structure in the **ShareInfo** member of the *InfoStruct* parameter. The **SHARE\_INFO\_0\_CONTAINER** structure contains an array of **SHARE\_INFO\_0** structures.

- `shi0_netname` MUST be set to `share.shi503_netname`.

If the **Level** member is 1, the server MUST return the information about **share** resources by filling the **SHARE\_INFO\_1\_CONTAINER** structure in the **ShareInfo** member of the *InfoStruct* parameter. The **SHARE\_INFO\_1\_CONTAINER** structure contains an array of **SHARE\_INFO\_1** structures.

- `shi1_netname` MUST be set to `share.shi503_netname`.
- `shi1_type` MUST be set to `share.shi503_type`.
- `shi1_remark` MUST be set to `share.shi503_remark`.

If the **Level** member is 2, the server MUST return the information about **share** resources by filling the **SHARE\_INFO\_2\_CONTAINER** structure in the **ShareInfo** member of the *InfoStruct* parameter. The **SHARE\_INFO\_2\_CONTAINER** structure contains an array of **SHARE\_INFO\_2** structures.

- `shi2_netname` MUST be set to `share.shi503_netname`.
- `shi2_type` MUST be set to `share.shi503_type`.
- `shi2_remark` MUST be set to `share.shi503_remark`.
- `shi2_permissions` MUST be set to `share.shi503_permissions`.
- `shi2_max_uses` MUST be set to `share.shi503_max_uses`.
- `shi2_current_uses` MUST be set to the sum of `share.shi503_current_uses` values retrieved from both CIFS and SMB2 servers.
- `shi2_path` MUST be set to `share.shi503_path`.
- `shi2_passwd` MUST be set to `share.shi503_passwd`.

If the **Level** member is 501, the server MUST return the information about **share** resources by filling the **SHARE\_INFO\_501\_CONTAINER** structure in the **ShareInfo** member of the *InfoStruct* parameter. The **SHARE\_INFO\_501\_CONTAINER** structure contains an array of **SHARE\_INFO\_501** structures.

- `shi501_netname` MUST be set to `share.shi503_netname`.
- `shi501_type` MUST be set to `share.shi503_type`.
- `shi501_remark` MUST be set to `share.shi503_remark`.
- `shi501_flags` MUST be set to `share.ShareFlags`.

If the **Level** member is 502, the server MUST return the information about **Share** resources by filling the `SHARE_INFO_502_CONTAINER` structure in the **ShareInfo** member of the *InfoStruct* parameter. The `SHARE_INFO_502_CONTAINER` structure contains an array of `SHARE_INFO_502_I` structures.

- `shi502_netname` MUST be set to `share.shi503_netname`.
- `shi502_type` MUST be set to `share.shi503_type`.
- `shi502_remark` MUST be set to `share.shi503_remark`.
- `shi502_permissions` MUST be set to `share.shi503_permissions`.
- `shi502_max_uses` MUST be set to `share.shi503_max_uses`.
- `shi502_current_uses` MUST be set to the sum of `share.shi503_current_uses` values retrieved from both CIFS and SMB2 servers.
- `shi502_path` MUST be set to `share.shi503_path`.
- `shi502_passwd` MUST be set to `share.shi503_passwd`.
- `shi502_security_descriptor` MUST be set to `share.shi503_security_descriptor`.

If the **Level** member is 503, the server MUST return the information about **share** resources in the `SHARE_INFO_503_I` structure by filling the `SHARE_INFO_503_CONTAINER` structure in the **ShareInfo** member of the *InfoStruct* parameter, except that `shi503_current_uses` MUST be set to the sum of `share.shi503_current_uses` values retrieved from both CIFS and SMB2 server. The `SHARE_INFO_503_CONTAINER` structure contains an array of `SHARE_INFO_503_I` structures.

The server MUST set the `STYPE_CLUSTER_FS`, `STYPE_CLUSTER_SOFS`, and `STYPE_CLUSTER_DFS` bits in the `shi*_type` field to zero; the client MUST ignore them on receipt.

The `PreferredMaximumLength` parameter specifies the maximum number of bytes that the server can return for the **ShareInfo** buffer. If `PreferredMaximumLength` is insufficient

to hold all the entries, the server MUST return the maximum number of entries that will fit in the **ShareInfo** buffer and return **ERROR\_MORE\_DATA**. If this parameter is equal to **MAX\_PREFERRED\_LENGTH** (section 2.2.2.2), the server MUST return all the requested data.

If the server returns **NERR\_Success** or **ERROR\_MORE\_DATA**, it MUST set the *TotalEntries* parameter to equal the total number of entries that could have been enumerated from the current resume position.

If *PreferredMaximumLength* is insufficient to hold all the entries and if the client has specified a *ResumeHandle*, the server MUST set *ResumeHandle* to some implementation-specific value that allows the server to continue with this enumeration on a subsequent call to this method with the same value for *ResumeHandle*.

The server MUST maintain the share list in the order in which shares are inserted into **ShareList**.

The following rules specify processing of the *ResumeHandle* parameter:

- If the *ResumeHandle* parameter is either **NULL** or points to **0x00000000**, the enumeration MUST start from the beginning of the **ShareList**.
- If the *ResumeHandle* parameter points to a nonzero value, the server MUST validate the *ResumeHandle*.
  - If the value of the *ResumeHandle* is less than the size of the **ShareList**, the server MUST continue enumeration based on the value of *ResumeHandle*. The value of *ResumeHandle* specifies the index into the **ShareList** after which enumeration is to begin.
  - If the value of the *ResumeHandle* is greater than or equal to the size of the **ShareList**, the server MUST return **NERR\_Success** and zero entries.
- If the client specified a *ResumeHandle* and if the server returns **ERROR\_MORE\_DATA** (**0x000000EA**), the server MUST set *ResumeHandle* to the index of the last enumerated share in the **ShareList**.

Because the *ResumeHandle* specifies the index into the **ShareList**, and the **ShareList** can be modified between multiple requests, the results of a query spanning multiple requests using the *ResumeHandle* can be unreliable, offering either duplicate or unavailable shares.

The server SHOULD [57](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [58](#) fail the call.

## 3.1.4.9 NetrShareEnumSticky (Opnum 36)

Article 10/30/2020

The NetrShareEnumSticky method retrieves information about each [sticky shared](#) resource whose IsPersistent setting is set in a [ShareList](#).

```
NET_API_STATUS NetrShareEnumSticky(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, out] LPSHARE_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**InfoStruct:** A pointer to a structure, in the format of a [SHARE\\_ENUM\\_STRUCT](#) (section 2.2.4.38). The [SHARE\\_ENUM\\_STRUCT](#) structure has a **Level** member that specifies the type of structure to return in the **ShareInfo** member. The **Level** member MUST be set to one of the values specified in section 2.2.4.38 (excluding [SHARE\\_INFO\\_501\\_CONTAINER](#)).

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of the returned data. If the specified value is MAX\_PREFERRED\_LENGTH, the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that could have been enumerated if the buffer had been big enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle, which is used to continue an existing [connection](#) search. The handle MUST be zero on the first call and remain unchanged for subsequent calls. If the *ResumeHandle* parameter is NULL, a resume handle MUST NOT be stored. If this parameter is not NULL and the method returns ERROR\_MORE\_DATA, this parameter receives an implementation-specific nonzero value that can be passed in subsequent calls to this method to continue with the enumeration.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the list of the currently active connections.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x000000EA ERROR_MORE_DATA	The client request succeeded. More entries are available. Not all entries could be returned in the buffer size that is specified by <i>PreferredMaximumLength</i> .
0x0000084B NERR_BufTooSmall	The client request succeeded. More entries are available. The buffer size that is specified by <i>PreferredMaximumLength</i> was too small to fit even a single entry.

In response to a NetrShareEnumSticky message, the server MUST enumerate all the sticky shares in the **ShareList** whose **IsPersistent** setting is set, or return an error code. If the server is restarted, any shares that are created before the restart that are not sticky MUST be forgotten. Information about sticky shares MUST be stored in a persistent store,<[59](#)> and the shares MUST be restored (that is, re-created on the server) after the server is restarted.

The NetrShareEnumSticky method MUST NOT support Level 501 and MUST enumerate only sticky shares. Other than this difference, the server MUST process this message in exactly the same manner as the [NetrShareEnum](#) message.

## 3.1.4.10 NetrShareGetInfo (Opnum 16)

Article 04/06/2021

The NetrShareGetInfo method retrieves information about a particular shared resource on the [server](#) from the [ShareList](#).

```
NET_API_STATUS NetrShareGetInfo(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* NetName,
    [in] DWORD Level,
    [out, switch_is(Level)] LPSHARE_INFO InfoStruct
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle ([\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**NetName:** A pointer to a null-terminated UTF-16 string that specifies the name of the share to return information for.

**Level:** Specifies the information level of the data. This parameter MUST be one of the following values.

Value	Meaning
0	<a href="#">LPSHARE_INFO_0</a>
1	<a href="#">LPSHARE_INFO_1</a>
2	<a href="#">LPSHARE_INFO_2</a>
501	<a href="#">LPSHARE_INFO_501</a>
502	<a href="#">LPSHARE_INFO_502_I</a>
503	<a href="#">LPSHARE_INFO_503_I</a>
1005	<a href="#">LPSHARE_INFO_1005</a>

**InfoStruct:** This parameter is of type [LPSHARE\\_INFO](#) union, as specified in section 2.2.3.6. Its contents are determined by the value of the *Level* parameter, as shown in the preceding table.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code

value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000084B NERR_BufTooSmall	The supplied buffer is too small.
0x00000906 NERR_NetNameNotFound	The share name does not exist.

If *ServerName* does not match any **Transport.ServerName** in **TransportList** with the **SVTI2\_SCOPED\_NAME** bit set in **Transport.Flags**, the server MUST reset *ServerName* as **"\*"**.

The server MUST remove any preceding "\\" from the parameter *ServerName* and normalize the *ServerName* parameter as specified in section 3.1.6.8, passing in the updated *ServerName* parameter as the *ServerName*, and an empty string as the *ShareName*.

The *NetName* parameter specifies the name of the share for which to return information. This MUST be a nonempty null-terminated UTF-16 string; otherwise, the server MUST fail the call with an **ERROR\_INVALID\_PARAMETER** error code.

The value of the *Level* parameter can be 0, 1, 2, 501, 502, 503, or 1005. If the value of the *Level* parameter is anything else, the server MUST fail the call with an

`ERROR_INVALID_LEVEL` error code. The value of the *Level* parameter determines the format of the *InfoStruct* parameter.

The server MUST locate a **Share** from **ShareList**, where *NetName* matches **Share.ShareName** and the normalized *ServerName* matches **Share.ServerName**. If no share is found, the server MUST fail the call with `NERR_NetNameNotFound` error code. If a matching **Share** is found, the server MUST query share properties 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, providing the tuple *<normalized server name, NetName>* as the input parameter. When the server receives `STATUS_SUCCESS` for a share, it MUST consider the received `SHARE_INFO_503_I` and `SHARE_INFO_1005` structures as valid. The server MUST return information about the shared resource on the server.

The server MUST use the **share** in valid `SHARE_INFO_503_I` and `SHARE_INFO_1005` structures from either CIFS or SMB2 servers and fill the return structures as follows. The server MUST discard the structures received from other file server except the value of `share.shi503_current_uses`.

If the value of the *Level* parameter is 0, the server MUST return information about the **share** by filling the `SHARE_INFO_0` structure in the **ShareInfo0** member of the *InfoStruct* parameter.

- `shi0_netname` MUST be set to `share.shi503_netname`.

If the value of the *Level* parameter is 1, the server MUST return information about the **share** by filling the `SHARE_INFO_1` structure in the **ShareInfo1** member of the *InfoStruct* parameter.

- `shi1_netname` MUST be set to `share.shi503_netname`.
- `shi1_type` MUST be set to `share.shi503_type`.
- `shi1_remark` MUST be set to `share.shi503_remark`.

If the value of the *Level* parameter is 2, the server MUST return information about the **share** by filling the `SHARE_INFO_2` structure in the **ShareInfo2** member of the *InfoStruct* parameter.

- `shi2_netname` MUST be set to `share.shi503_netname`.
- `shi2_type` MUST be set to `share.shi503_type`.
- `shi2_remark` MUST be set to `share.shi503_remark`.

- **shi2\_permissions** MUST be set to **share.shi503\_permissions**.
- **shi2\_max\_uses** MUST be set to **share.shi503\_max\_uses**.
- **shi2\_current\_uses** MUST be set to the sum of **share.shi503\_current\_uses** values retrieved from both CIFS and SMB2 servers.
- **shi2\_path** MUST be set to **share.shi503\_path**.
- **shi2\_passwd** MUST be set to **share.shi503\_passwd**.

If the value of the *Level* parameter is 501, the server MUST return information about the **share** by filling the **SHARE\_INFO\_501** structure in the **ShareInfo501** member of the *InfoStruct* parameter.

- **shi501\_netname** MUST be set to **share.shi503\_netname**.
- **shi501\_type** MUST be set to **share.shi503\_type**.
- **shi501\_remark** MUST be set to **share.shi503\_remark**.
- **shi501\_flags** MUST be set to **share.ShareFlags**.

If the value of the *Level* parameter is 502, the server MUST return information about the **share** by filling the **SHARE\_INFO\_502\_I** structure in the **ShareInfo502** member of the *InfoStruct* parameter.

- **shi502\_netname** MUST be set to **share.shi503\_netname**.
- **shi502\_type** MUST be set to **share.shi503\_type**.
- **shi502\_remark** MUST be set to **share.shi503\_remark**.
- **shi502\_permissions** MUST be set to **share.shi503\_permissions**.
- **shi502\_max\_uses** MUST be set to **share.shi503\_max\_uses**.
- **shi502\_current\_uses** MUST be set to the sum of **share.shi503\_current\_uses** values retrieved from both CIFS and SMB2 servers.
- **shi502\_path** MUST be set to **share.shi503\_path**.
- **shi502\_passwd** MUST be set to **share.shi503\_passwd**.
- **shi502\_security\_descriptor** MUST be set to **share.shi503\_security\_descriptor**.

If the value of the *Level* parameter is 503, the server MUST return information about the **share** in the **SHARE\_INFO\_503\_I** structure by filling the **SHARE\_INFO\_503\_I** structure in

the **ShareInfo503** member of the *InfoStruct* parameter, except that **shi503\_current\_uses** MUST be set to the sum of **share.shi503\_current\_uses** values retrieved from both CIFS and SMB2 servers.

The server MUST set the **STYPE\_CLUSTER\_FS**, **STYPE\_CLUSTER\_SOFS**, and **STYPE\_CLUSTER\_DFS** bits of the **shi\*\_type** field to zero; the client MUST ignore them on receipt.

If the value of the *Level* parameter is 1005, the server MUST return information about the **share** in the **SHARE\_INFO\_1005** structure directly by filling the **SHARE\_INFO\_1005** structure in the **ShareInfo1005** member of the *InfoStruct* parameter.

If both the SMB server and the SMB2 server return an error, the server MUST fail the call with **NERR\_NetNameNotFound** error code.

The server SHOULD<60> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<61> fail the call.

## 3.1.4.11 NetrShareSetInfo (Opnum 17)

Article 06/24/2021

The NetrShareSetInfo method sets the parameters of a shared resource in a ShareList.

```
NET_API_STATUS NetrShareSetInfo(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* NetName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSHARE_INFO ShareInfo,
    [in, out, unique] DWORD* ParmErr
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle ([\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**NetName:** A pointer to a null-terminated UTF-16 string that specifies the name of the [share](#) to set information for.

**Level:** Specifies the information level of the data. This parameter MUST be one of the following values.

Value	Meaning
1	<a href="#">LPSHARE_INFO_1</a>
2	<a href="#">LPSHARE_INFO_2</a>
502	<a href="#">SHARE_INFO_502_I</a>
503	<a href="#">SHARE_INFO_503_I</a>
1004	<a href="#">LPSHARE_INFO_1004</a>
1005	<a href="#">LPSHARE_INFO_1005</a>
1006	<a href="#">LPSHARE_INFO_1006</a>
1501	<a href="#">LPSHARE_INFO_1501_I</a>

**ShareInfo:** This parameter is of type [LPSHARE\\_INFO](#) union, as specified in section 2.2.3.6. Its contents are determined by the value of the *Level* parameter, as shown in the preceding table. This parameter MUST NOT contain a null value. If the *Level* parameter is equal to 1, 2, 502, or 503, the client MUST set the [STYPE\\_CLUSTER\\_FS](#),

`STYPE_CLUSTER_SOFS`, and `STYPE_CLUSTER_DFS` bits in the `shi*``_type` field of the `SHARE_INFO` union to zero; the server MUST ignore them on receipt.

**ParmErr:** A pointer to a value that receives the index of the first member of the share information structure that caused the `ERROR_INVALID_PARAMETER` error, if it occurs.

**Return Values:** The method returns `0x00000000` (`NERR_Success`) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
<code>0x00000000</code> <code>NERR_Success</code>	The client request succeeded.
<code>0x00000005</code> <code>ERROR_ACCESS_DENIED</code>	Access is denied.
<code>0x00000057</code> <code>ERROR_INVALID_PARAMETER</code>	The client request failed because the specified parameter is invalid. For details, see the description that follows for the <i>ParmErr</i> parameter.
<code>0x00000008</code> <code>ERROR_NOT_ENOUGH_MEMORY</code>	Not enough storage is available to process this command.
<code>0x00000906</code> <code>NERR_NetNameNotFound</code>	The share name does not exist.
<code>0x00000032</code> <code>ERROR_NOT_SUPPORTED</code>	The server does not support branch cache. < <a href="#">62</a> >
<code>0x00000424</code> <code>ERROR_SERVICE_DOES_NOT_EXIST</code>	The branch cache component does not exist as an installed service. < <a href="#">63</a> >
<code>0x0000007C</code> <code>ERROR_INVALID_LEVEL</code>	The system call level is not correct.

In response to a `NetrShareSetInfo` message, the server MUST set the parameters of a shared resource or return an error code.

The *NetName* parameter specifies the name of the share for which to set information in `ShareList`. The *NetName* MUST be a nonempty, null-terminated UTF-16 string;

otherwise, the server MUST fail the call with an **ERROR\_INVALID\_PARAMETER** error code.

The value of the *Level* parameter can be 1, 2, 502, 503, 1004, 1005, 1006, or 1501. If the value of the *Level* parameter is anything else, the server MUST fail the call with an **ERROR\_INVALID\_LEVEL** error code. The value of the *Level* parameter determines the format of the *InfoStruct* parameter.

If *ServerName* does not match any **Transport.ServerName** in **TransportList** with the **SVTI2\_SCOPED\_NAME** bit set in **Transport.Flags**, the server MUST reset *ServerName* as **"\*"**.

The server MUST remove any preceding \\ from the *ServerName* parameter and normalize the *ServerName* parameter as specified in section [3.1.6.8](#), passing in the updated *ServerName* parameter as the *ServerName*, and an empty string as the *ShareName*.

The server MUST validate all information that is provided in the **SHARE\_INFO** structure. If a member of the **SHARE\_INFO** structure is found to be invalid, the server MUST fail the call with an **ERROR\_INVALID\_PARAMETER** error code. The server does the following validation on the **SHARE\_INFO** structure:

- If *shi\*\_type* has the flag **STYPE\_SPECIAL**, a security descriptor MUST NOT be specified in *shi502\_security\_descriptor* (*Level* = 502).
- If *shi\*\_remark* is specified, its length MUST NOT be greater than 48.
- If *Level*=502 and a security descriptor is provided, it MUST be a valid security descriptor.

If the *ParmErr* parameter is not NULL and the server finds a member of the **SHARE\_INFO** structure to be invalid, the server MUST set *ParmErr* to a value that denotes the index of the member that was found to have an invalid value and fail the call with **ERROR\_INVALID\_PARAMETER** (0x00000057). The mapping between the values to set and the corresponding member MUST be as specified in section [2.2.2.11](#).

The server MUST locate a **Share** from **ShareList**, where *NetName* matches **Share.ShareName** and *ServerName* matches **Share.ServerName**. If no share is found, the server MUST fail the call with a **NERR\_NetNameNotFound** error code.

If a matching share is found, the server MUST construct a **SHARE\_INFO\_503\_I** structure and a **SHARE\_INFO\_1005** structure from the share, as specified in section [3.1.3](#).

The server MUST update the members of **SHARE\_INFO\_503\_I** and **SHARE\_INFO\_1005** structures based on the *Level* parameter, as follows:

If the *Level* parameter is equal to 1, all the settings that are defined by the SHARE\_INFO\_1 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrShareSetInfo method) MUST be updated. The share properties MUST be updated as follows:

- **SHARE\_INFO\_503\_I.shi503\_remark** MUST be set to **shi1\_remark**.

If the *Level* parameter is equal to 2, all the settings that are defined by the SHARE\_INFO\_2 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrShareSetInfo method) MUST be updated. The share properties MUST be updated as follows:

- **SHARE\_INFO\_503\_I.shi503\_remark** MUST be set to **shi2\_remark**.
- **SHARE\_INFO\_503\_I.shi503\_max\_uses** MUST be set to **shi2\_max\_uses**.

If the *Level* parameter is equal to 502, all the settings that are defined by the SHARE\_INFO\_502\_I structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrShareSetInfo method) MUST be updated. The share properties MUST be updated as follows:

- **SHARE\_INFO\_503\_I.shi503\_remark** MUST be set to **shi502\_remark**.
- **SHARE\_INFO\_503\_I.shi503\_max\_uses** MUST be set to **shi502\_max\_uses**.
- **SHARE\_INFO\_503\_I.shi503\_security\_descriptor** MUST be set to **shi502\_security\_descriptor**.

If the *Level* parameter is equal to 503, all the settings that are defined by the SHARE\_INFO\_503\_I structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrShareSetInfo method) MUST be updated. The share properties MUST be updated as follows:

- **SHARE\_INFO\_503\_I.shi503\_remark** MUST be set to **shi503\_remark**.
- **SHARE\_INFO\_503\_I.shi503\_max\_uses** MUST be set to **shi503\_max\_uses**.
- **SHARE\_INFO\_503\_I.shi503\_security\_descriptor** MUST be set to **shi503\_security\_descriptor**.

If the *Level* parameter is equal to 1004, all the settings that are defined by the SHARE\_INFO\_1004 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrShareSetInfo method) MUST be updated.

- **SHARE\_INFO\_503\_I.shi503\_remark** MUST be set to **shi1004\_remark**.

If the *Level* parameter is equal to 1005, all the settings that are defined by the **SHARE\_INFO\_1005** structure as settable (that is, they are not defined as ignored on receipt or ignored for the **NetrShareSetInfo** method) MUST be updated. Only disk shares can be affected by this *Level*. The share MUST be updated as follows:<64>

- **SHARE\_INFO\_1005.shi1005\_flags** MUST be set to **shi1005\_flags**.

If the *Level* parameter is equal to 1006, all the settings that are defined by the **SHARE\_INFO\_1006** structure as settable (that is, they are not defined as ignored on receipt or ignored for the **NetrShareSetInfo** method) MUST be updated. The share properties are updated as follows:

- **SHARE\_INFO\_503\_I.shi503\_max\_uses** MUST be set to **shi1006\_max\_uses**.

If the *Level* parameter is equal to 1501, all the settings that are defined by the **SHARE\_INFO\_1501\_I** structure as settable (that is, they are not defined as ignored on receipt or ignored for the **NetrShareSetInfo** method) MUST be updated. The share properties MUST be updated as follows:

- **SHARE\_INFO\_503\_I.shi503\_security\_descriptor** MUST be set to **shi1501\_security\_descriptor**.

The server MUST invoke the underlying server events as specified in [\[MS-CIFS\]](#) section 3.3.4.10 or [\[MS-SMB\]](#) section 3.3.4.6 and [\[MS-SMB2\]](#) section 3.3.4.14, providing the updated **SHARE\_INFO\_503\_I** structure and the updated **SHARE\_INFO\_1005** structure as input parameters.

If both the SMB and SMB2 servers return an error, the server MUST fail the call with **ERROR\_INVALID\_DATA**.

If only one of the SMB and SMB2 servers returns **STATUS\_SUCCESS**:

- The server MUST construct a new **SHARE\_INFO\_503\_I** structure and a new **SHARE\_INFO\_1005** structure from the **Share**, as specified in section 3.1.3.
- The server MUST revert the updates made to the share on the server that returned **STATUS\_SUCCESS** by invoking the underlying server event (as specified in [\[MS-CIFS\]](#) section 3.3.4.10, [\[MS-SMB\]](#) section 3.3.4.6, or [\[MS-SMB2\]](#) section 3.3.4.14), providing the **SHARE\_INFO\_503\_I** structure and the **SHARE\_INFO\_1005** structure as input parameters.
- The server MUST return **ERROR\_INVALID\_DATA** to the caller.

If both the SMB and the SMB2 servers return **STATUS\_SUCCESS**, the server MUST update the **Share** as follows and return **NERR\_Success** to the caller:

- If the *Level* parameter is equal to 1, 2, 502, 503, or 1004, **Share.Remark** MUST be set to *shi\*\_remark*.
- If the *Level* parameter is equal to 2, 502, 503, or 1006, **Share.MaxUses** MUST be set to *shi\*\_max\_uses*.
- If the *Level* parameter is equal to 502, 503, or 1501, **Share.FileSecurity** MUST be set to *shi\*\_security\_descriptor* if *Level* is equal to 502 or 503; otherwise, it MUST be set to NULL.
- If the *Level* parameter is equal to 1005:
  - **Share.CscFlags** MUST be set to the value of *shi1005\_flags* masked by **CSC\_MASK** as specified in section 2.2.4.29.
  - **Share.IsDfs** MUST be set to TRUE if *shi1005\_flags* contains **SHI1005\_FLAGS\_DFS** or **SHI1005\_FLAGS\_ROOT** as specified in section 2.2.4.29; otherwise, it MUST be set to FALSE.
  - **Share.DoAccessBasedDirectoryEnumeration** MUST be set to TRUE if *shi1005\_flags* contains **SHI1005\_FLAGS\_ACCESS\_BASED\_DIRECTORY\_ENUM** bit as specified in section 2.2.4.29; otherwise it MUST be set to FALSE.
  - **Share.AllowNamespaceCaching** MUST be set to True if *shi1005\_flags* contains **SHI1005\_FLAGS\_ALLOW\_NAMESPACE\_CACHING** bit as specified in section 2.2.4.29; otherwise, it MUST be set to FALSE.
  - **Share.ForceSharedDelete** MUST be set to TRUE if *shi1005\_flags* contains **SHI1005\_FLAGS\_FORCE\_SHARED\_DELETE** bit as specified in section 2.2.4.29; otherwise, it MUST be set to FALSE.
  - **Share.RestrictExclusiveOpens** MUST be set to TRUE if *shi1005\_flags* contains **SHI1005\_FLAGS\_RESTRICT\_EXCLUSIVE\_OPENS** bit as specified in section 2.2.4.29; otherwise, it MUST be set to FALSE.
  - **Share.HashEnabled** MUST be set to TRUE if *shi1005\_flags* contains **SHI1005\_FLAGS\_ENABLE\_HASH** bit as specified in section 2.2.4.29; otherwise it MUST be set to FALSE.
  - **Share.ForceLevel2Oplock** MUST be set to TRUE if *shi1005\_flags* contains **SHI1005\_FLAGS\_FORCE\_LEVELII\_OPLOCK** bit as specified in section 2.2.4.29; otherwise, it MUST be set to FALSE.

The server SHOULD [65](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required

credentials, the server SHOULD<sup>66</sup> fail the call.

## 3.1.4.12 NetrShareDel (Opnum 18)

Article 10/30/2020

The NetrShareDel method deletes a share name from the **ShareList**, which disconnects all [connections](#) to the shared resource. If the share is [sticky](#), all information about the share is also deleted from permanent storage.<67>

```
NET_API_STATUS NetrShareDel(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* NetName,
    [in] DWORD Reserved
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle ([\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**NetName:** A pointer to a null-terminated UTF-16 string that specifies the name of the share to delete.

**Reserved:** The server MUST ignore this parameter.<68>

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

Return value/code	Description
0x00000906	The share name does not exist.
NERR_NetNameNotFound	

If *ServerName* does not match any **Transport.ServerName** in **TransportList** with the SVTI2\_SCOPED\_NAME bit set in **Transport.Flags**, the server MUST reset *ServerName* as "*\**".

The server MUST remove any preceding "\\\" from the *ServerName* parameter and normalize the *ServerName* parameter as specified in section 3.1.6.8, passing in the updated *ServerName* parameter as the *ServerName*, and an empty string as the *ShareName*.

The server MUST look up the **ShareList** and locate a **Share** where *NetName* matches **Share.ShareName** and *ServerName* matches **Share.ServerName**. If no match is found, the server MUST fail the call with a NERR\_NetNameNotFound error code. If a matching share is found, the server MUST remove the share from **ShareList** and free the share object.

If the Share is found and **Share.IsPrinterShare** is TRUE, **PrinterShareCount** MUST be decreased by 1. If **PrinterShareCount** becomes 0, the server MUST invoke the events as specified in section 3.1.6.10, providing SV\_TYPE\_PRINTQ\_SERVER as input parameter.

The server MUST delete the Share by invoking underlying server event as specified in [MS-CIFS] section 3.3.4.11 and [MS-SMB2] section 3.3.4.15, providing tuple <*ServerName*, *NetName*> as input parameters. If either CIFS or SMB2 servers return STATUS\_SUCCESS, the server MUST return NERR\_Success. Otherwise, the server MUST fail the call with an implementation-dependent error.

The server SHOULD<69> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<70> fail the call.

## 3.1.4.13 NetrShareDelSticky (Opnum 19)

Article 02/14/2019

The NetrShareDelSticky method marks the share as nonpersistent by clearing the `IsPersistent` member of a Share in the `ShareList`.

```
NET_API_STATUS NetrShareDelSticky(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* NetName,
    [in] DWORD Reserved
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the `server`. The `client` MUST map this structure to an RPC binding handle ([\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**NetName:** A pointer to a null-terminated UTF-16 string that specifies the name of the share to delete.

**Reserved:** The server MUST ignore this parameter.[<71>](#)

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

The primary use of this method is to delete a sticky share whose root directory has been deleted (thus preventing actual re-creation of the share) but whose entry still exists in permanent storage.[<72>](#) This method can also be used to remove the persistence of a share without deleting the current incarnation of the share.

If `ServerName` does not match any `Transport.ServerName` in `TransportList` with the `SVTI2_SCOPED_NAME` bit set in `Transport.Flags`, the server MUST reset `ServerName` as `"*"`.

The server MUST remove any preceding "\\" from the `ServerName` parameter and normalize the `ServerName` parameter as specified in section 3.1.6.8, passing in the updated `ServerName` parameter as the `ServerName`, and an empty string as the `ShareName`.

The `NetName` parameter specifies the name of the share to delete. This MUST be a nonempty, null-terminated UTF-16 string; otherwise, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The server MUST search through **ShareList** and locate a **Share** where **Share.ShareName** matches *NetName*, **Share.ServerName** matches *ServerName*, and **Share.IsPersistent** is TRUE. If a match is not found, the server MUST fail the call with an NERR\_NetNameNotFound error code.

If a match is found, the server MUST make the share nonpersistent by setting **Share.IsPersistent** to FALSE and the server MUST delete the share entry from permanent storage.

The server SHOULD<sup><73></sup> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<sup><74></sup> fail the call.

## 3.1.4.14 NetrShareDelStart (Opnum 37)

Article 06/24/2021

The NetrShareDelStart method performs the initial phase of a two-phase share delete.

```
NET_API_STATUS NetrShareDelStart(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* NetName,
    [in] DWORD Reserved,
    [out] PSHARE_DEL_HANDLE ContextHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an [RPC](#) binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**NetName:** A pointer to a null-terminated UTF-16 string that specifies the name of the share to delete.

**Reserved:** Reserved; SHOULD be set to zero when sent and MUST be ignored on receipt.

**ContextHandle:** A handle for the second phase of the two-phase share delete, in the form of a [PSHARE\\_DEL\\_HANDLE](#) (section 2.2.1.3) data type.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrShareDelStart request, the server MUST mark a share for deletion and return to the client an RPC context handle that the client can use to actually perform the deletion by calling the [NetrShareDelCommit](#) method.

This two-phase deletion MUST be used to delete IPC\$, which is the share that is used for [named pipes](#). Deleting IPC\$ results in the closing of the pipe on which the RPC is being executed. Thus, the client never receives the response to the RPC. The two-phase delete offers a positive response in phase 1 and then an expected error in phase 2.

If *ServerName* does not match any [Transport.ServerName](#) in [TransportList](#) with the SVTI2\_SCOPED\_NAME bit set in [Transport.Flags](#), the server MUST reset *ServerName* as "/\*".

The server MUST remove any preceding "\\" from the *ServerName* parameter and normalize the *ServerName* parameter as specified in section 3.1.6.8, passing in the

updated *ServerName* parameter as the *ServerName*, and an empty string as the *ShareName*.

The server MUST search through **ShareList** and locate a **Share** where **Share.ShareName** matches **NetName** and **Share.ServerName** matches *ServerName*. If a match is not found, the server MUST fail the call with an NERR\_NetNameNotFound error code.

If a match is found, the server MUST mark the share for deletion by setting the **IsMarkedForDeletion** member of the **Share** element in **ShareList**. The share MUST remain available until the client calls the **NetrShareDelCommit** method.

The server MUST return a handle to the share being deleted in the *ContextHandle* parameter. The client is expected to use the handle to actually delete the share by calling the **NetrShareDelCommit** method.

The server SHOULD<75> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<76> fail the call.

## 3.1.4.15 NetrShareDelCommit (Opnum 38)

Article 06/24/2021

The NetrShareDelCommit method performs the final phase of a two-phase share delete.

```
NET_API_STATUS NetrShareDelCommit(
    [in, out] PSHARE_DEL_HANDLE ContextHandle
);
```

**ContextHandle:** A handle returned by the first phase of a two-phase share delete.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success. Otherwise, the method returns a nonzero error code unless the share being deleted is IPC\$. If the share being deleted is IPC\$, the return value is not meaningful. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

The NetrShareDelCommit message is the continuation of the [NetrShareDelStart](#) message and MUST cause the share to be actually deleted, which disconnects all [connections](#) to the share, or MUST return an error code.

This method can be used to delete the IPC\$ share as well as other shares. When the share is not IPC\$, only a return value of 0 indicates success.

This two-phase deletion MUST be used to delete IPC\$, which is the share that is used for [named pipes](#). Deleting IPC\$ results in the closing of the pipe on which the [RPC](#) is being executed. Thus, the [client](#) never receives the response to the RPC. The two-phase delete offers a positive response in phase 1 and then an expected error in phase 2.

**ContextHandle** MUST reference the share to be deleted in the [NetrShareDelStart](#) method. If a share is not found, the [server](#) MUST fail the call with an [ERROR\\_INVALID\\_PARAMETER](#) error code.

If a share is found, but the [IsMarkedForDeletion](#) member of the [Share](#) is not set, the server MUST fail the call with an [ERROR\\_INVALID\\_PARAMETER](#) error code.

Otherwise, the server MUST delete the share by invoking the underlying server event, as specified in [\[MS-CIFS\]](#) section 3.3.4.11 and [\[MS-SMB2\]](#) section 3.3.4.15, providing tuple <ServerName, NetName> as input parameters.

The server does not enforce any security measures when processing this call.

## 3.1.4.16 NetrShareCheck (Opnum 20)

Article 06/24/2021

The NetrShareCheck method checks whether a [server](#) is sharing a device.

```
NET_API_STATUS NetrShareCheck(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* Device,
    [out] DWORD* Type
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) ([section 2.2.1.1](#)) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) [sections 4.3.5](#) and [5.1.5.2](#)). The server MUST ignore this parameter.

**Device:** A pointer to a null-terminated UTF-16 string that specifies the name of the device to check for shared access.

**Type:** A pointer to a DWORD that receives the type of the shared device. This parameter is set only if the method returns successfully. On success, the server MUST set this parameter as specified in [section 2.2.2.4](#), except that STYPE\_SPECIAL is not returned.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#). The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000907 NERR_DeviceNotShared	The device is not shared.

In response to a NetrShareCheck request, the server MUST scan through the [ShareList](#). For each [share](#), if [Share.LocalPath](#), as specified in [\[MS-SMB2\]](#) section [3.3.1.6](#) or [\[MS-CIFS\]](#) section [3.3.1.2](#), points to the device or volume specified by the caller, the server MUST return the type of the matching device in the *Type* parameter. The type can be

one of the values that are listed in Share Types (section 2.2.2.4). In response to a NetrShareCheck message, the server MUST check whether it is sharing a device and return a response to the client.

The *Device* parameter specifies the name of the shared device to check for. The server MUST enumerate the active shared devices, and if it finds a match to the *Device* parameter, the server MUST return the type of the matching device in the *Type* parameter. The type can be one of the values that are listed in Share Types. The server MUST set the STYPE\_CLUSTER\_FS, STYPE\_CLUSTER\_SOFS, and STYPE\_CLUSTER\_DFS bits of the *Type* parameter to zero; the client MUST ignore them on receipt.

If no match is found, the server MUST fail the call by using an NERR\_DeviceNotShared error code.

The server does not enforce any security measures when it processes this call.

## 3.1.4.17 NetrServerGetInfo (Opnum 21)

Article 06/24/2021

The NetrServerGetInfo method retrieves current configuration information for CIFS and SMB Version 1.0 servers.

```
NET_API_STATUS NetrServerGetInfo(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [out, switch_is(Level)] LPSERVER_INFO InfoStruct
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the *server*. The *client* MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2).

**Level:** Specifies the information level of the data. The value of the *Level* parameter determines the contents of the *InfoStruct* parameter. This parameter MUST be one of the following values.

Value	Meaning
100	<a href="#">LPERVER_INFO_100</a>
101	<a href="#">LPERVER_INFO_101</a>
102	<a href="#">LPERVER_INFO_102</a>
103	<a href="#">LPERVER_INFO_103</a>
502	<a href="#">LPERVER_INFO_502</a>
503	<a href="#">LPERVER_INFO_503</a>

**InfoStruct:** This is a structure of type [LPERVER\\_INFO](#), as specified in section 2.2.3.7. The content of the *InfoStruct* parameter is determined by the *Level* parameter, as the preceding table shows.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
-------------------	-------------

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

In response to the `NetrServerGetInfo` request, the server MUST return configuration information from the `ServerConfiguration` object based on the value of the *Level* parameter.

The value of the *Level* parameter can be 100, 101, 102, 502, or 503. If the *Level* parameter has any other value, the server MUST fail the call with an `ERROR_INVALID_LEVEL` error code.<77>

The value of the *Level* parameter determines the format of the *InfoStruct* parameter.

If the value of the *Level* parameter is 100, the server MUST return its information by filling the `SERVER_INFO_100` structure in the `ServerInfo100` member of the *InfoStruct* parameter.

- `sv100_platform_id` MUST be set to `ServerConfiguration.ServerInfo103.sv103_platform_id`.
- If the `ServerName` parameter is NULL, `sv100_name` MUST be set to `ServerConfiguration.ServerInfo103.sv103_name`. Otherwise, `sv100_name` MUST be set to the value of `ServerName`.

If the value of the *Level* parameter is 101, the server MUST return its information by filling the `SERVER_INFO_101` structure in the `ServerInfo101` member of the *InfoStruct* parameter.

- sv101\_platform\_id MUST be set to `ServerConfiguration.ServerInfo103.sv103_platform_id`.
- If the *ServerName* parameter is NULL, sv101\_name MUST be set to `ServerConfiguration.ServerInfo103.sv103_name`. Otherwise, sv101\_name MUST be set to the value of *ServerName*.
- sv101\_sv101\_version\_major MUST be set to `ServerConfiguration.ServerInfo103.sv103_version_major`.
- sv101\_version\_minor MUST be set to `ServerConfiguration.ServerInfo103.sv103_version_minor`.
- sv101\_type MUST be set to **GlobalServerAnnounce**.
- sv101\_comment MUST be set to `ServerConfiguration.ServerInfo103.sv103_comment`.

If the value of the *Level* parameter is 102, the server MUST return its information by filling the SERVER\_INFO\_102 structure in the ServerInfo102 member of the *InfoStruct* parameter.

- sv102\_platform\_id MUST be set to `ServerConfiguration.ServerInfo103.sv103_platform_id`.
- If the *ServerName* parameter is NULL, sv102\_name MUST be set to `ServerConfiguration.ServerInfo103.sv103_name`. Otherwise, sv102\_name MUST be set to the value of *ServerName*.
- sv102\_version\_major MUST be set to `ServerConfiguration.ServerInfo103.sv103_version_major`.
- sv102\_version\_minor MUST be set to `ServerConfiguration.ServerInfo103.sv103_version_minor`.
- sv102\_type MUST be set to **GlobalServerAnnounce**.
- sv102\_comment MUST be set to `ServerConfiguration.ServerInfo103.sv103_comment`.
- sv102\_users MUST be set to `ServerConfiguration.ServerInfo103.sv103_users`.
- sv102\_disc MUST be set to `ServerConfiguration.ServerInfo103.sv103_disc`.
- sv102\_hidden MUST be set to `ServerConfiguration.ServerInfo103.sv103_hidden`.

- **sv102\_anndelta** MUST be set to  
`ServerConfiguration.ServerInfo103.sv103_anndelta`.
- **sv102\_licenses** MUST be set to 0.

If the value of the *Level* parameter is 103, the server MUST return server information in `ServerConfiguration.ServerInfo103` directly by filling the SERVER\_INFO\_103 structure in the `ServerInfo103` member of the `InfoStruct` parameter and setting `sv103_type` to `GlobalServerAnnounce`.[<78>](#)

If the value of the *Level* parameter is 502, the server MUST return its information by filling the SERVER\_INFO\_502 structure in the `ServerInfo502` member of the `InfoStruct` parameter.

- **sv502\_sessopens** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_sessopens`.
- **sv502\_sessvcs** MUST be set to `ServerConfiguration.ServerInfo599.sv599_sessvcs`.
- **sv502\_opensearch** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_opensearch`.
- **sv502\_sizreqbuf** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_sizreqbuf`.
- **sv502\_initworkitems** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_initworkitems`.
- **sv502\_maxworkitems** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxworkitems`.
- **sv502\_rawworkitems** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_rawworkitems`.
- **sv502\_irpstacksize** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_irpstacksize`.
- **sv502\_maxrawbuflen** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxrawbuflen`.
- **sv502\_sessusers** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_sessusers`.
- **sv502\_sessconns** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_sessconns`.

- **sv502\_maxpagedmemoryusage** MUST be set to `ServerConfiguration.ServerInfo599.sv599_maxpagedmemoryusage`.
- **sv502\_maxnonpagedmemoryusage** MUST be set to `ServerConfiguration.ServerInfo599.sv599_maxnonpagedmemoryusage`.
- **sv502\_enablessoftcompat** MUST be set to `ServerConfiguration.ServerInfo599.sv599_enablessoftcompat`.
- **sv502\_enableforcedlogoff** MUST be set to `ServerConfiguration.ServerInfo599.sv599_enableforcedlogoff`.
- **sv502\_timesource** MUST be set to `ServerConfiguration.ServerInfo599.sv599_timesource`.
- **sv502\_acceptdownlevelapis** MUST be set to `ServerConfiguration.ServerInfo599.sv599_acceptdownlevelapis`.
- **sv502\_lmannounce** MUST be set to `ServerConfiguration.ServerInfo599.sv599_lmannounce`.

If the value of the *Level* parameter is 503, the server MUST return its information by filling the SERVER\_INFO\_503 structure in the ServerInfo503 member of the *InfoStruct* parameter.

- **sv503\_sessopens** MUST be set to `ServerConfiguration.ServerInfo599.sv599_sessopens`.
- **sv503\_sessvcs** MUST be set to `ServerConfiguration.ServerInfo599.sv599_sessvcs`.
- **sv503\_opensearch** MUST be set to `ServerConfiguration.ServerInfo599.sv599_opensearch`.
- **sv503\_sizreqbuf** MUST be set to `ServerConfiguration.ServerInfo599.sv599_sizreqbuf`.
- **sv503\_initworkitems** MUST be set to `ServerConfiguration.ServerInfo599.sv599_initworkitems`.
- **sv503\_maxworkitems** MUST be set to `ServerConfiguration.ServerInfo599.sv599_maxworkitems`.
- **sv503\_rawworkitems** MUST be set to `ServerConfiguration.ServerInfo599.sv599_rawworkitems`.

- **sv503\_irpstacksize** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_irpstacksize.`
- **sv503\_maxrawbuflen** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxrawbuflen.`
- **sv503\_sessusers** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_sessusers.`
- **sv503\_sessconns** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_sessconns.`
- **sv503\_maxpagedmemoryusage** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxpagedmemoryusage.`
- **sv503\_maxnonpagedmemoryusage** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxnonpagedmemoryusage.`
- **sv503\_enablessoftcompat** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enablessoftcompat.`
- **sv503\_enableforcedlogoff** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enableforcedlogoff.`
- **sv503\_timesource** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_timesource.`
- **sv503\_acceptdownlevelapis** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_acceptdownlevelapis.`
- **sv503\_lmannounce** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_lmannounce.`
- **sv503\_domain** MUST be set to `ServerConfiguration.ServerInfo599.sv599_domain.`
- **sv503\_maxcopyreadlen** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxcopyreadlen.`
- **sv503\_maxcopywritelen** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxcopywritelen.`
- **sv503\_minkeepsearch** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_minkeepsearch.`
- **sv503\_maxkeepsearch** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxkeepsearch.`

- **sv503\_minkeepcomplsearch** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_minkeepcomplsearch.`
- **sv503\_maxkeepcomplsearch** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxkeepcomplsearch.`
- **sv503\_threadcountadd** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_threadcountadd.`
- **sv503\_numblockthreads** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_numblockthreads.`
- **sv503\_scavtimeout** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_scavtimeout.`
- **sv503\_minrcvqueue** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_minrcvqueue.`
- **sv503\_minfreeworkitems** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_minfreeworkitems.`
- **sv503\_xactmemsize** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_xactmemsize.`
- **sv503\_threadpriority** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_threadpriority.`
- **sv503\_maxmpxct** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxmpxct.`
- **sv503\_oplockbreakwait** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_oplockbreakwait.`
- **sv503\_oplockbreakresponsewait** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_oplockbreakresponsewait.`
- **sv503\_enableoplocks** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enableoplocks.`
- **sv503\_enableoplockforceclose** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enableoplockforceclose.`
- **sv503\_enablefcbopens** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enablefcbopens.`

- **sv503\_enableraw** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enableraw.`
- **sv503\_enablesharednetdrives** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_enablesharednetdrives.`
- **sv503\_minfreeconnections** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_minfreeconnections.`
- **sv503\_maxfreeconnections** MUST be set to  
`ServerConfiguration.ServerInfo599.sv599_maxfreeconnections.`

The server SHOULD [79](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [80](#) fail the call.

The *ServerName* parameter MUST be either NULL or a null-terminated string, as described in section 2.2.1.1. If it is non-NULL, the length of the string MUST be less than 1,024 or the server MUST fail the call with `ERROR_INVALID_PARAMETER`.

## 3.1.4.18 NetrServerSetInfo (Opnum 22)

Article 06/24/2021

The NetrServerSetInfo method sets [server](#) operating parameters for CIFS and SMB Version 1.0 file servers; it can set them individually or collectively. The information is stored in a way that allows it to remain in effect after the system is reinitialized. <81>

```
NET_API_STATUS NetrServerSetInfo(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSERVER_INFO ServerInfo,
    [in, out, unique] DWORD* ParmErr
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. The value of the *Level* parameter determines the contents of the *ServerInfo* parameter. This parameter MUST be one of the values in the following table. The NetrServerSetInfo method does not support a *Level* value of 103. If a *Level* value of 103 is specified, the server MUST return [ERROR\\_INVALID\\_LEVEL](#).

Value	Meaning
101	<a href="#">LPERVER_INFO_101</a>
102	<a href="#">LPERVER_INFO_102</a>
502	<a href="#">LPERVER_INFO_502</a>
503	<a href="#">LPERVER_INFO_503</a>
599	<a href="#">LPERVER_INFO_599</a>
1005	<a href="#">LPERVER_INFO_1005</a>
1107	<a href="#">LPERVER_INFO_1107</a>
1010	<a href="#">LPERVER_INFO_1010</a>
1016	<a href="#">LPERVER_INFO_1016</a>
1017	<a href="#">LPERVER_INFO_1017</a>

Value	Meaning
1018	LPSERVER_INFO_1018
1501	LPSERVER_INFO_1501
1502	LPSERVER_INFO_1502
1503	LPSERVER_INFO_1503
1506	LPSERVER_INFO_1506
1510	LPSERVER_INFO_1510
1511	LPSERVER_INFO_1511
1512	LPSERVER_INFO_1512
1513	LPSERVER_INFO_1513
1514	LPSERVER_INFO_1514
1515	LPSERVER_INFO_1515
1516	LPSERVER_INFO_1516
1518	LPSERVER_INFO_1518
1523	LPSERVER_INFO_1523
1528	LPSERVER_INFO_1528
1529	LPSERVER_INFO_1529
1530	LPSERVER_INFO_1530
1533	LPSERVER_INFO_1533
1534	LPSERVER_INFO_1534
1535	LPSERVER_INFO_1535
1536	LPSERVER_INFO_1536
1538	LPSERVER_INFO_1538
1539	LPSERVER_INFO_1539
1540	LPSERVER_INFO_1540
1541	LPSERVER_INFO_1541
1542	LPSERVER_INFO_1542

Value	Meaning
1543	LPSERVER_INFO_1543
1544	LPSERVER_INFO_1544
1545	LPSERVER_INFO_1545
1546	LPSERVER_INFO_1546
1547	LPSERVER_INFO_1547
1548	LPSERVER_INFO_1548
1549	LPSERVER_INFO_1549
1550	LPSERVER_INFO_1550
1552	LPSERVER_INFO_1552
1553	LPSERVER_INFO_1553
1554	LPSERVER_INFO_1554
1555	LPSERVER_INFO_1555
1556	LPSERVER_INFO_1556

**ServerInfo:** This is a structure of type LPSERVER\_INFO, as specified in section [2.2.3.7](#). The content of the *ServerInfo* parameter is determined by the *Level* parameter, as the preceding table shows.

**ParmErr:** A pointer to a value that receives the index of the first member of the server information structure that caused an ERROR\_INVALID\_PARAMETER error code, if it occurs.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#). The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Return value/code	Description
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid. For details see the description that follows for the <i>ParmErr</i> parameter.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

In response to a *NetrServerSetInfo* request, the server MUST update the **ServerConfiguration** object based on the caller-supplied values and the *Level*. The server can set its operating parameters individually or collectively. The information is stored in a way that allows it to remain in effect after the system is reinitialized.

The value of the *Level* parameter can be 101, 102, 502, 503, 599, 1005, 1107, 1010, 1016, 1017, 1018, 1501, 1502, 1503, 1506, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1518, 1523, 1528, 1529, 1530, 1533, 1534, 1535, 1536, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1552, 1553, 1554, 1555, and 1556.

As previously stated, a *Level* value of 103 is not supported by the *NetrServerSetInfo* method. If the *Level* parameter has any other value, the server MUST fail the call with an **ERROR\_INVALID\_LEVEL** error code.

After receiving the *NetrServerSetInfo* method, the server MUST update the server setting that corresponds to the *ServerInfo* parameter. The format for the *ServerInfo* parameter is as specified in **SERVER\_INFO** (section 2.2.3.7).

If the *Level* parameter is equal to 101, the server MUST update all the settings in **ServerConfiguration.ServerInfo103** that are defined by the **SERVER\_INFO\_101** structure as settable (that is, they are not defined as ignored on receipt or ignored for the *NetrServerSetInfo* method).

If the *Level* parameter is equal to 102, the server MUST update all the settings in **ServerConfiguration.ServerInfo103** that are defined by the **SERVER\_INFO\_102** structure as settable (that is, they are not defined as ignored on receipt or ignored for the *NetrServerSetInfo* method).

If the *Level* parameter is equal to 502, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the **SERVER\_INFO\_502** structure as settable (that is, they are not defined as ignored on receipt or ignored for the *NetrServerSetInfo* method).

If the *Level* parameter is equal to 503, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_503 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 599, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_599 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1005, the server MUST update all the settings in **ServerConfiguration** that are defined by the SERVER\_INFO\_1005 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1107, the server MUST update all the settings in **ServerConfiguration.ServerInfo103** that are defined by the SERVER\_INFO\_1107 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1016, the server MUST update all the settings in **ServerConfiguration.ServerInfo103** that are defined by the SERVER\_INFO\_1016 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1017, the server MUST update all the settings in **ServerConfiguration.ServerInfo103** that are defined by the SERVER\_INFO\_1017 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1018, the server MUST update all the settings in **ServerConfiguration.ServerInfo103** that are defined by the SERVER\_INFO\_1018 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1501, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1501 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1502, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1502

structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1503, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1503 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1506, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1506 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1510, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1510 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1511, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1511 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1512, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1512 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1513, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1513 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1514, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1514 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1515, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1515 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1516, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1516 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1518, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1518 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1523, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1523 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1528, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1528 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1529, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1529 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1530, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1530 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1533, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1533 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1534, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1534 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1535, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1535

structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1536, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1536 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1538, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1538 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1539, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1539 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1540, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1540 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1541, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1541 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1542, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1542 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1543, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1543 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1544, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1544 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1545, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1545 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1546, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1546 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1547, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1547 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1548, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1548 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1549, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1549 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1550, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1550 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1552, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1552 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1553, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1553 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1554, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1554

structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1555, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1555 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

If the *Level* parameter is equal to 1556, the server MUST update all the settings in **ServerConfiguration.ServerInfo599** that are defined by the SERVER\_INFO\_1556 structure as settable (that is, they are not defined as ignored on receipt or ignored for the NetrServerSetInfo method).

The server MUST validate each member of the structure that is passed in the *ServerInfo* parameter. The validation involves making sure each member of the structure in the *ServerInfo* parameter has a valid value as specified in the definition of the corresponding SERVER\_INFO structure. If any member of the structure is not valid and the *ParmErr* parameter is not NULL, the server MUST set *ParmErr* to a value based on the first member of the structure that is not valid and fail the call with an ERROR\_INVALID\_PARAMETER (0x00000057) error code. The mapping between the values to set and the corresponding member is listed in section [2.2.2.12.<82>](#)

The server MUST construct SERVER\_INFO\_103 and SERVER\_INFO\_599 structures from **ServerConfiguration.ServerInfo103** and **ServerConfiguration.ServerInfo599** respectively.

The server MUST update server configuration by invoking the underlying server event as specified in [MS-CIFS] section [3.3.4.22](#), providing SERVER\_INFO\_103 and SERVER\_INFO\_599 structures as input parameters.

The server MUST update browser configuration by invoking the underlying server event specified in [MS-BRWS] section [3.2.4.1](#), providing the SERVER\_INFO\_103 structure as input parameter.

The server MUST persist the values in **ServerConfiguration.ServerInfo103** and **ServerConfiguration.ServerInfo599** in a persistent configuration store.

The server SHOULD [<83>](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [<84>](#) fail the call.

## 3.1.4.19 NetrServerDiskEnum (Opnum 23)

Article 10/30/2020

The NetrServerDiskEnum method retrieves a list of disk drives on a [server](#). The method returns an array of three-character strings (a drive letter, a colon, and a terminating null character).

```
NET_API_STATUS NetrServerDiskEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, out] DISK_ENUM_CONTAINER* DiskInfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. It MUST be the following value.

Value	Meaning
0	The buffer is of type <a href="#">DISK_INFO</a> .

**DiskInfoStruct:** A pointer to a structure of type [DISK\\_ENUM\\_CONTAINER](#), as specified in section 2.2.4.92. Although this parameter is defined as an [in, out] parameter, it is used only as an [out] parameter. The server MUST ignore any values that are passed in this parameter.

**PreferredMaximumLength:** The server MUST ignore this parameter.

**TotalEntries:** The number of entries being returned in the **Buffer** member of the *DiskInfoStruct* parameter. This MUST be in the range 0–26.

**ResumeHandle:** The server MUST ignore this parameter.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000005 ERROR_ACCESS_DENIED	The caller does not have the permissions to perform the operation.

The server MUST ignore the *PreferredMaximumLength* parameter.

The server MUST ignore the *ResumeHandle* parameter.

Upon successful processing of the request, the server MUST set the *TotalEntries* parameter equal to the number of disk drive entries that the server enumerated in the **Buffer** member of *DiskInfoStruct* and the **EntriesRead** member of *DiskInfoStruct* MUST be set to 1 plus the value set for *TotalEntries*.

Upon successful processing of the request, the server MUST return the enumerated disk drives in the **Buffer** member of *DiskInfoStruct* in the format of the **DISK\_INFO** structure. The server MUST allocate the memory required to return all enumerated disk drives in the **Buffer** member of the *InfoStruct* parameter. In cases where the RPC allocated a buffer because the client specified a non-NULL value for the *Buffer* parameter, the server MUST free the buffer that is allocated by the [RPC](#).

The server SHOULD [85](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [86](#) fail the call.

## 3.1.4.20 NetrServerStatisticsGet (Opnum 24)

Article 02/14/2019

The NetrServerStatisticsGet method retrieves the operating statistics for a service.

```
NET_API_STATUS NetrServerStatisticsGet(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* Service,
    [in] DWORD Level,
    [in] DWORD Options,
    [out] LPSTAT_SERVER_0* InfoStruct
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Service:** A pointer to a null-terminated UTF-16 string. This parameter MUST be ignored on receipt.

**Level:** Specifies the information level of the data. This MUST be set to 0.

**Options:** Reserved; MUST be 0.

**InfoStruct:** A pointer to the buffer that receives the data, as specified in section [2.2.4.39](#). This pointer is in the format of [STAT\\_SERVER\\_0](#).

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#).

In response to the NetrServerStatisticsGet message, the server MUST return the operating statistics for the service or return an error code.

The server MUST ignore the *Service* parameter on receipt.

If the *Level* parameter is not equal to 0, the server MUST fail the call with an [ERROR\\_INVALID\\_LEVEL](#) error code.

If the *Options* parameter is not equal to 0, the server MUST fail the call with an [ERROR\\_INVALID\\_PARAMETER](#) error code.

The server MUST query the statistics by invoking the underlying server events as specified in [MS-CIFS] section 3.3.4.23 and [MS-SMB2] section 3.3.4.24. The server MUST aggregate all the values in the structures received from both CIFS and SMB2 servers into a new **STAT\_SERVER\_0** structure. In addition to these values, *sts0\_start* MUST be set to **StatisticsStartTime**. The server MUST return the statistics in the **STAT\_SERVER\_0** structure in the **InfoStruct** parameter.

The server SHOULD<sup>87</sup> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<sup>88</sup> fail the call.

## 3.1.4.21 NetrRemoteTOD (Opnum 28)

Article02/14/2019

The NetrRemoteTOD method returns the time of day information on a [server](#).

```
NET_API_STATUS NetrRemoteTOD(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [out] LPTIME_OF_DAY_INFO* BufferPtr
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**BufferPtr:** A pointer to a structure of type [TIME\\_OF\\_DAY\\_INFO](#) where the information is returned.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrRemoteTOD message, the server MUST return the time of day information or return an error code.

The server MUST return the time of day information on the server in the *BufferPtr* parameter in the format of the LPTIME\_OF\_DAY\_INFO structure, as specified in section 2.2.4.105.

The server SHOULD [<89>](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [<90>](#) fail the call.

## 3.1.4.22 NetrServerTransportAdd (Opnum 25)

Article 10/30/2020

The NetrServerTransportAdd method binds the [server](#) to the transport protocol.

```
NET_API_STATUS NetrServerTransportAdd(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in] LP.getServer_TRANSPORT_INFO_0 Buffer
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. This parameter MUST be zero.

**Buffer:** A pointer to the [SERVER\\_TRANSPORT\\_INFO\\_0](#) structure that describes the data.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000034 ERROR_DUP_NAME	A duplicate name exists on the network.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Return value/code	Description
0x00000008	Not enough storage is available to process this command.
ERROR_NOT_ENOUGH_MEMORY	

The NetrServerTransportAdd message MUST be processed in the same way as the [NetrServerTransportAddEx](#) message, except that it MUST allow only level 0 (that is, *SERVER\_TRANSPORT\_INFO\_0*). The NetrServerTransportAddEx message is specified in section 3.1.4.23.

The server MAY [\*\*<91>\*\*](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD [\*\*<92>\*\*](#) fail the call.

## 3.1.4.23 NetrServerTransportAddEx (Opnum 41)

Article 10/30/2020

The `NetrServerTransportAddEx` method binds the specified `server` to the transport protocol. This extended method allows the caller to specify information levels 1, 2, and 3 beyond what the `NetrServerTransportAdd` method allows.

```
NET_API_STATUS NetrServerTransportAddEx(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPTRANSPORT_INFO Buffer
);
```

**ServerName:** An `SRVSVC_HANDLE` (section 2.2.1.1) pointer that identifies the server. The `client` MUST map this structure to an RPC binding handle (see [C706] ↗ sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. This parameter MUST be the following value.

Value	Meaning
0	The buffer is of type <code>SERVER_TRANSPORT_INFO_0</code> .
1	The buffer is of type <code>SERVER_TRANSPORT_INFO_1</code> .
2	The buffer is of type <code>SERVER_TRANSPORT_INFO_2</code> .
3	The buffer is of type <code>SERVER_TRANSPORT_INFO_3</code> .

**Buffer:** A pointer to the `TRANSPORT_INFO` union that describes the data. The type of data depends on the value of the *Level* parameter, as the preceding table shows.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
-------------------	-------------

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000034 ERROR_DUP_NAME	A duplicate name exists on the network.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

The server SHOULD [93](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD [94](#) fail the call.

The *Level* parameter determines the type of structure that the client has used to specify information about the new transport. The value MUST be 0, 1, 2, or 3. If the *Level* parameter is not equal to one of the valid values, the server MUST fail the call with an ERROR\_INVALID\_LEVEL error code.

If the *Level* parameter is 0, the *Buffer* parameter points to a SERVER\_TRANSPORT\_INFO\_0 structure.

If the *Level* parameter is 1, the *Buffer* parameter points to a SERVER\_TRANSPORT\_INFO\_1 structure.

If the *Level* parameter is 2, the *Buffer* parameter points to a SERVER\_TRANSPORT\_INFO\_2 structure.

If the *Level* parameter is 3, the *Buffer* parameter points to a SERVER\_TRANSPORT\_INFO\_3 structure.

The server MUST validate all information that is provided in the SERVER\_TRANSPORT\_INFO structure and MUST fail the call with

ERROR\_INVALID\_PARAMETER if any of these checks fail:

- Both *svti\*\_transportname* and *svti\*\_transportaddress* MUST NOT be NULL; *svti\*\_transportaddresslength* MUST NOT be zero.
- If *svti\*\_domain* is not NULL, its length MUST NOT be greater than 15.
- The *svti\*\_flags* can be any combination of the following flags as defined in section 2.2.4.96: 0, SVTI2\_REMAP\_PIPE NAMES, and SVTI2\_SCOPED\_NAME.

The server MUST invoke the events specified in [MS-CIFS] section 3.3.4.17 and [MS-SMB2] section 3.3.4.21, passing the following as the parameters: *svti\*\_transportname*, *svti\*\_transportaddress*, and a transport enable flag set to TRUE.

If both the CIFS and SMB2 servers return ERROR\_NOT\_SUPPORTED, the server MUST return ERROR\_NOT\_SUPPORTED (0x00000032) to the caller. If both the CIFS and SMB2 servers return an error other than ERROR\_NOT\_SUPPORTED, the server must fail the call with an implementation-dependent error.

If either the CIFS or SMB2 server returns STATUS\_SUCCESS, the server MUST create a new Transport and add it to the **TransportList**. The Transport MUST be initialized as follows:

- **Transport.Name** MUST be set to the caller-supplied *svti\*\_transportname*. For acceptable forms of *svti\*\_transportname*, see section 2.2.4.96.
- **Transport.ServerName** MUST be set to the caller-supplied *svti\*\_transportaddress*. For acceptable forms of *svti\*\_transportaddress*, see section 2.2.4.96.
- **Transport.Domain** MUST be set to *svti\*\_domain*.
- **Transport.Flags** MUST be set to *svti\*\_flags*.
- **Transport.ConnectionCount** MUST be set to zero.
- The Transport MUST be persisted in an implementation-specific store.

The server MUST then return NERR\_Success to the caller.

## 3.1.4.24 NetrServerTransportEnum (Opnum 26)

Article 10/04/2021

The NetrServerTransportEnum method enumerates the information about transport protocols that the [server](#) manages in [TransportList](#).

```
NET_API_STATUS NetrServerTransportEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, out] LP SERVER_XPORT_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD* TotalEntries,
    [in, out, unique] DWORD* ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**InfoStruct:** A pointer to a structure, in the format of a [SERVER\\_XPORT\\_ENUM\\_STRUCT](#) structure that receives the data. The [SERVER\\_XPORT\\_ENUM\\_STRUCT](#) structure has a **Level** member that specifies the type of the structure to return in the [XportInfo](#) member. The **Level** member MUST be set to one of the values in section 2.2.4.101 (excluding [SERVER\\_XPORT\\_INFO\\_3\\_CONTAINER](#)).

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of returned data. If the value that is specified is [MAX\\_PREFERRED\\_LENGTH](#) (section 2.2.2.2), the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that can be enumerated if the buffer is large enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle that is used to continue an existing [connection](#) search. The handle MUST be zero on the first call and remain unchanged for subsequent calls. If the *ResumeHandle* parameter is NULL, no resume handle MUST be stored. If this parameter is not NULL and the method returns [ERROR\\_MORE\\_DATA](#), this parameter receives an implementation-specific nonzero value that can be passed in subsequent calls to this method to continue with the enumeration.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the list of the currently active connections.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The client request succeeded.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x000000EA ERROR_MORE_DATA	The client request succeeded. More entries are available. Not all entries could be returned in the buffer size that is specified by <i>PreferredMaximumLength</i> .
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000084B NERR_BufTooSmall	The client request succeeded. More entries are available. The buffer size that is specified by <i>PreferredMaximumLength</i> was too small to fit even a single entry.

In response to the NetrServerTransportEnum request, the server MUST enumerate the Transports from the **TransportList** or return an error code.

The **InfoStruct** parameter has a **Level** member. The value of *Level* MUST be 0, 1, or 2. If the **Level** member is not equal to one of the valid values, the server MUST fail the call with an **ERROR\_INVALID\_LEVEL** error code.

If the value of the **Level** member is 0, the server MUST return the information about the transport protocols that it is managing by filling the **SERVER\_XPORT\_INFO\_0\_CONTAINER** structure in the **XportInfo** member of the **InfoStruct** parameter.

If the **Level** member is 1, the server MUST return the information about the transport protocols that it is managing by filling the **SERVER\_XPORT\_INFO\_1\_CONTAINER** structure in the **XportInfo** member of the **InfoStruct** parameter.

The *PreferredMaximumLength* parameter specifies the maximum number of bytes that the server can return for the **XportInfo** buffer.

If the *PreferredMaximumLength* is insufficient to hold all the entries, the server MUST return the maximum number of entries that can fit in the **XportInfo** buffer and return

`ERROR_MORE_DATA`. If this parameter is equal to `MAX_PREFERRED_LENGTH`, the server MUST return all the requested data.

If the server returns `NERR_Success` or `ERROR_MORE_DATA`, it MUST set the `TotalEntries` parameter equal to the total number of entries that could have been enumerated from the current resume position.

If the `PreferredMaximumLength` is insufficient to hold all the entries and if the client has specified a `ResumeHandle` parameter, the server MUST set `ResumeHandle` to some implementation-specific value that allows the server to continue with this enumeration on a subsequent call to this method with the same value for `ResumeHandle`.

The following rules specify processing of the `ResumeHandle` parameter:

- If the `ResumeHandle` parameter is either `NULL` or points to `0x00000000`, the enumeration MUST start from the beginning of the `TransportList`.
- If the `ResumeHandle` parameter points to a nonzero value, the server MUST continue enumeration based on the value of `ResumeHandle`. The value of `ResumeHandle` specifies the index into the `TransportList` after which the enumeration is to begin.
- If the client specified a `ResumeHandle` and if the server returns `ERROR_MORE_DATA` (`0x000000EA`), the server MUST set `ResumeHandle` to the index of the last enumerated transport in the `TransportList`.

Because the `ResumeHandle` parameter specifies an offset into the list, and the list of all available transports can be modified between multiple requests, the results of a query spanning multiple requests using `ResumeHandle` can be unreliable, offering either duplicate or unavailable transports.

The server SHOULD<sup>95</sup> enforce security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD<sup>96</sup> fail the call.

## 3.1.4.25 NetrServerTransportDel (Opnum 27)

Article 06/24/2021

The `NetrServerTransportDel` method unbinds (or disconnects) the transport protocol from the [server](#). If this method succeeds, the server can no longer communicate with [clients](#) by using the specified transport protocol (such as TCP or XNS).

```
NET_API_STATUS NetrServerTransportDel(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in] LPSERVER_TRANSPORT_INFO_0 Buffer
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. This SHOULD be zero and MUST be ignored on receipt.

Value	Meaning
0	The buffer is of type <a href="#">SERVER_TRANSPORT_INFO_0</a> .

**Buffer:** A pointer to the [SERVER\\_TRANSPORT\\_INFO\\_0](#) structure that contains information about the transport.

**Return Values:** The method returns 0x00000000 ([NERR\\_Success](#)) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000	The client request succeeded.
NERR_Success	
0x00000005	Access is denied.
ERROR_ACCESS_DENIED	

Return value/code	Description
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

The `NetrServerTransportDel` message MUST be processed in the same way as the [NetrServerTransportDelEx](#) message, except that it MUST allow only level 0 (that is, `SERVER_TRANSPORT_INFO_0`). The processing for this message is specified in section 3.1.4.26.

The server MAY [97](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD [98](#) fail the call.

## 3.1.4.26 NetrServerTransportDelEx (Opnum 53)

Article 10/04/2021

The [server](#) receives the NetrServerTransportDelEx method in an RPC\_REQUEST packet. In response, the server unbinds (or disconnects) the transport protocol from the server. If this method succeeds, the server can no longer communicate with [clients](#) by using the specified transport protocol (such as TCP or XNS). This extended method allows level 1 beyond what the [NetrServerTransportDel](#) method allows.

```
NET_API_STATUS NetrServerTransportDelEx(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPTRANSPORT_INFO Buffer
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. It MUST be one of the following values.

Value	Meaning
0	The buffer is of type <a href="#">SERVER_XPORT_INFO_0_CONTAINER</a> .
1	The buffer is of type <a href="#">SERVER_XPORT_INFO_1_CONTAINER</a> .

**Buffer:** A pointer to the [TRANSPORT\\_INFO](#) union that contains information about the transport. The value of the *Level* parameter determines the type of the contents of the *Buffer* parameter, as the preceding table shows.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.<99>

The *Level* parameter determines the type of structure the client has used to specify information about the new transport. Valid values are 0 and 1. If the *Level* parameter is not equal to one of the valid values, the server MUST fail the call with an [ERROR\\_INVALID\\_LEVEL](#) error code.

If the *Level* parameter is 0, the *Buffer* parameter points to a [SERVER\\_TRANSPORT\\_INFO\\_0](#) structure. If the *Level* parameter is 1, the *Buffer* parameter points to a [SERVER\\_TRANSPORT\\_INFO\\_1](#) structure.

The server MUST validate all information that is provided in the [SERVER\\_TRANSPORT\\_INFO](#) structure in an implementation-specific manner, and, if any member of the structure is found to be invalid, the server MUST fail the call with an [ERROR\\_INVALID\\_PARAMETER](#) error code.

The server MUST look up the Transport in the [TransportList](#), where [Transport.Name](#) matches the caller-supplied *svti\*\_transportname* and [Transport.ServerName](#) matches the caller-supplied *svti\*\_transportaddress*. If a match is not found, the server MUST return [NERR\\_NetNameNotFound](#) to the caller.

If a match is found, the server MUST invoke the events described in [\[MS-CIFS\]](#) section [3.3.4.17](#) and [\[MS-SMB2\]](#) section [3.3.4.21](#), passing *Transport.ServerName*, *Transport.Name*, and a transport enable flag set to FALSE as the parameters. This means that the SMB file server can no longer initiate communications with clients by using the specified transport protocol (such as SMB2 over Direct TCP).<100>

If both the CIFS and SMB2 servers return [ERROR\\_NOT\\_SUPPORTED](#), the server MUST return [ERROR\\_NOT\\_SUPPORTED](#) (0x00000032) to the caller. If both the CIFS and SMB2 servers return an error other than [ERROR\\_NOT\\_SUPPORTED](#), the server must fail the call with an implementation-dependent error.

If either the CIFS or SMB2 server returns [STATUS\\_SUCCESS](#), the server MUST remove [Transport](#) from [TransportList](#) and from the persistent store, free the transport object and return [NERR\\_Success](#).

The server SHOULD<101> enforce security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD<102> fail the call.

## 3.1.4.27 NetrpGetFileSecurity (Opnum 39)

Article04/06/2021

The NetrpGetFileSecurity method returns to the caller a copy of the security descriptor that protects a file or directory.

```
DWORD NetrpGetFileSecurity(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* ShareName,
    [in, string] WCHAR* lpFileName,
    [in] SECURITY_INFORMATION RequestedInformation,
    [out] PADT_SECURITY_DESCRIPTOR* SecurityDescriptor
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the *server*. The *client* MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**ShareName:** A pointer to a null-terminated UTF-16 string that specifies the share name on which the file is found.

**lpFileName:** A pointer to a null-terminated UTF-16 string that specifies the name of the file or directory whose security is being retrieved. The name MUST specify the full path to the file from the *ShareName* parameter.

**RequestedInformation:** The type of security information being requested, as specified in [\[MS-DTYP\]](#) section 2.4.7.

**SecurityDescriptor:** A pointer to a [PADT\\_SECURITY\\_DESCRIPTOR](#) structure, where the desired information is returned.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrpGetFileSecurity message, the server MUST return to the caller a copy of the security descriptor that protects a file or directory, or return an error code. The security descriptor is always returned in the self-relative format.

The *ShareName* parameter specifies a local share name on the server. The server MUST locate a **Share** from **ShareList**, where *ShareName* matches **Share.ShareName**. If no share is found, the server MUST fail the call with NERR\_NetNameNotFound. The server MUST

then combine **Share.LocalPath** with the *lpFileName* parameter in order to create a fully qualified path name that is local to the server. If the file does not exist, the server SHOULD<103> fail the call with ERROR\_FILE\_NOT\_FOUND. The server MUST then obtain the security descriptor with the information that the client requires, as specified in the *RequestedInformation* parameter, for the local file that the path name obtained specifies, and return it to the client in the out parameter *SecurityDescriptor*. The security descriptor itself is stored in the **Buffer** member of the *SecurityDescriptor* parameter; the length of the security descriptor is stored in the **Length** member.

The server SHOULD<104> enforce security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD<105> fail the call.

## 3.1.4.28 NetrpSetFileSecurity (Opnum 40)

Article 06/24/2021

The NetrpSetFileSecurity method sets the security of a file or directory.

```
DWORD NetrpSetFileSecurity(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string, unique] WCHAR* ShareName,
    [in, string] WCHAR* lpFileName,
    [in] SECURITY_INFORMATION SecurityInformation,
    [in] PADT_SECURITY_DESCRIPTOR SecurityDescriptor
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**ShareName:** A pointer to a null-terminated UTF-16 string that specifies the share name on which the file is found.

**lpFileName:** A pointer to a null-terminated UTF-16 string that specifies the name of the file or directory whose security is being set.

**SecurityInformation:** The type of security information being set, as specified in [\[MS-DTYP\]](#) section 2.4.7.

**SecurityDescriptor:** A pointer to a [PADT\\_SECURITY\\_DESCRIPTOR](#) structure, which provides the security descriptor to set.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrpSetFileSecurity message, the server MUST set the security descriptor of the specified file or directory on the server or return an error code.

The *ShareName* parameter specifies a local share name on the server. The server MUST locate a **Share** from **ShareList**, where *ShareName* matches **Share.ShareName**. If no share is found, the server MUST fail the call with NERR\_NetNameNotFound. The server MUST then combine **Share.LocalPath** with the *lpFileName* parameter to create a fully qualified path name that is local to the server. If the file does not exist, the server MUST fail the call with ERROR\_FILE\_NOT\_FOUND.

The *SecurityDescriptor* parameter has a **Buffer** member that contains a security descriptor in self-relative format and a **Length** member that specifies the length, in bytes, of the **Buffer** member. The server MUST apply the descriptor in the **Buffer** member to the local file, whose *PathName* was computed as previously specified, by combining the local path that corresponds to the *ShareName* parameter and the *lpFileName* parameter.

The server SHOULD<106> enforce security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD<107> fail the call.

## 3.1.4.29 NetprPathType (Opnum 30)

Article 10/30/2020

The NetprPathType method checks a path name to determine its type.

```
NET_API_STATUS NetprPathType(  
    [in, string, unique] SRVSVC_HANDLE ServerName,  
    [in, string] WCHAR* PathName,  
    [out] DWORD* PathType,  
    [in] DWORD Flags  
) ;
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**PathName:** A pointer to a null-terminated UTF-16 string that specifies the path name to check.

**PathType:** A path type is returned. It MUST be one of the values that are defined in section [2.2.2.9](#).

**Flags:** A bitmask that MUST contain the bitwise OR of zero or more of the following values specifying controlling flags.

Value	Meaning
0x00000001	If set, the method uses old-style path rules (128-byte paths, 8.3 components) when validating the path. This flag is set on MS-DOS and OS/2 1.1 systems.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#).

In response to a NetprPathType message, the server MUST parse the specified path, determining if it is a valid path and determining its path type, or return an error code. Path type values are defined in section 2.2.2.9.

The *PathName* parameter specifies the path name whose type needs to be determined.

If the *PathName* parameter is an empty string or has a length greater than 260, the server MUST fail the call with ERROR\_INVALID\_NAME. If the *Flag* parameter has a value other than 0 or 1, the server MUST fail the call with ERROR\_INVALID\_PARAMETER.

If the *Flag* parameter is 0x1, the server MUST use old (MS-DOS) style path name rules that state that a path name can be 128 bytes long and that the file portion of the path has an 8-bit name and a 3-bit extension. If the value of the *Flag* parameter is 0x0, the server MUST use the long path name rules as specified in [MS-CIFS] section 2.2.1.1.1.

The server MUST obtain the path type value for the PathName by using the algorithm as specified in section 3.1.1.9. If the algorithm yields ERROR\_INVALID\_NAME, the server MUST fail the call with the same error code. Otherwise, the server MUST copy the path type value resulting from the algorithm into PathType and return NERR\_Success.

The server MAY<sup><108></sup> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<sup><109></sup> fail the call.

## 3.1.4.30 NetprPathCanonicalize (Opnum 31)

Article 06/24/2021

The NetprPathCanonicalize method converts a path name to the canonical format.

```
NET_API_STATUS NetprPathCanonicalize(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* PathName,
    [out, size_is(OutbufLen)] unsigned char* Outbuf,
    [in, range(0,64000)] DWORD OutbufLen,
    [in, string] WCHAR* Prefix,
    [in, out] DWORD* PathType,
    [in] DWORD Flags
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an [RPC](#) binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**PathName:** A pointer to a null-terminated UTF-16 string that specifies the path name to canonicalize.

**Outbuf:** A pointer to the output buffer where the canonicalized path name is returned.

**OutbufLen:** The length, in bytes, of the output buffer, *Outbuf*. The value of this field MUST be within the range 0–64,000, inclusive.

**Prefix:** A pointer to a null-terminated UTF-16 string that specifies an optional prefix to use when canonicalizing a relative path name.

**PathType:** A place to store the path type. This parameter MUST be set by the client either to zero or to one of the values defined in section [2.2.2.9](#). After successful completion of the request, the server MUST set *PathType* to one of the values defined in section 2.2.2.9.

**Flags:** Reserved, MUST be zero.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#).

If the *Flags* parameter is not equal to zero, the server SHOULD fail the call with an implementation-specific error code.<110>

In response to a NetprPathCanonicalize message, the server MUST compute the canonical version of the specified path name or return an error code.

The *PathName* parameter specifies the path name that needs to be canonicalized.

The *PathType* parameter, if nonzero, MUST specify the path type of the path that is specified by the *PathName* parameter by a previous successful call to the [NetprPathType](#) method. Even if it is set to the correct nonzero value by the client, the server can change it because the canonicalized version of a name can be of a different type than the original version. If *PathType* is zero, the server MUST validate and get the type of *PathName* (as specified in section 3.1.4.29) first. If this fails, the server MUST fail the call with an ERROR\_INVALID\_NAME error code.

The *Prefix* parameter, if it is a nonempty string, specifies a path component that MUST be prefixed to *PathName* to get the full path to canonicalize. The server MUST treat *Prefix* as a *PathName*: it MUST validate and get the type of *Prefix* in the same way as it does the *PathName*. If this fails, the server MUST fail the call with an ERROR\_INVALID\_NAME error code. The optional *Prefix* parameter is a convenience that this method provides to clients. The client is free to construct the complete *PathName* and pass NULL for the *Prefix*. For example, this parameter can be used when canonicalizing path names for a list of files in a directory. In such a scenario, the value for *Prefix* is the absolute path for the directory, and the value for *PathName* specifies the relative path for a file.

The *OutBufLen* parameter specifies the length of the output buffer *OutBuf* that is provided by the client. If the length of the canonicalized path name is greater than *OutBufLen*, the server MUST fail the call with an NERR\_BufTooSmall error code.

The server MUST construct the path to canonicalize by appending the *PathName* to the *Prefix*. If the *Prefix* parameter does not end with one, the server SHOULD insert an implementation-specific path separator between the *Prefix* and *PathName*.<111> The server MUST then canonicalize the resultant path. The canonicalization process is implementation-dependent.<112>

After the canonicalization is successfully finished, the server MUST determine the path type of the canonicalized path name, as specified in [NetprPathType](#) (section 3.1.4.29), and store the result in the *PathType* parameter. Valid return codes for the *PathType* parameter are as specified in Path Types (section 2.2.2.9). If this fails, the server MUST fail the call with an ERROR\_INVALID\_NAME error code.

The server MAY<113> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<114> fail the call.

### 3.1.4.31 NetprPathCompare (Opnum 32)

Article 06/24/2021

The NetprPathCompare method performs comparison of two paths.

```
long NetprPathCompare(
    [in, string, unique] SRV SVC _HANDLE ServerName,
    [in, string] WCHAR* PathName1,
    [in, string] WCHAR* PathName2,
    [in] DWORD PathType,
    [in] DWORD Flags
);
```

**ServerName:** An [SRV SVC \\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**PathName1:** A pointer to a null-terminated UTF-16 string that contains the first PathName to compare.

**PathName2:** A pointer to a null-terminated UTF-16 string that contains the second PathName to compare.

**PathType:** The type of PathName, as specified in section [2.2.2.9](#).

**Flags:** A bitmask that MUST contain the bitwise OR of zero or more of the following values that specify controlling flags.

Value	Meaning
0x00000001	SHOULD be set if both of the paths have already been canonicalized.

**Return Values:** Upon successful processing, the server MUST return 0 if both paths are the same, -1 if the first is less than the second, and 1 otherwise. If the method fails, it can return any specific error code value as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetprPathCompare message, the server MUST compare the two paths that are specified as parameters to see if they match and return this result or return an error code. If the supplied names are not canonicalized, the server MUST do the canonicalization of the path names before a comparison can occur. This does not modify the input path names. The clients SHOULD call this method with canonicalized path names only, because the canonicalization operation can be expensive. If

uncanonicalized path names are passed in, the caller SHOULD be aware that a nonzero result could be due to an error that occurred during canonicalization.

The *PathName1* and *PathName2* parameters specify the two path names to be compared.

The *Flags* parameter MUST be either 0 or 1. If the *Flags* parameter has any other value, the server MUST fail the call with ERROR\_INVALID\_PARAMETER. If the *Flags* parameter is 1, it implies that the specified path names are already canonicalized and the server MUST not try to canonicalize them.

Any combination of *Name1* (canonicalized or not), *Name2* (canonicalized or not), and *Flags* (0 or 1) is valid.

If *Flags* is set to 0, the server MUST first attempt to canonicalize both *Name1* and *Name2* (and MUST respond with an error if canonicalization fails) before comparing the names.

If *Flags* is set to 1, the server MUST compare the names without first attempting canonicalization. Using Flags=1 could optimize performance because it eliminates the need for the server to repeatedly canonicalize a path name if it is being compared multiple times. If the *Flags* parameter does not have a valid value, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

If the *Flags* parameter is 1, the *PathType* parameter MUST specify the path type for the two path names. Valid values for the *PathType* parameter are as specified in section 2.2.2.9. If the *PathType* parameter does not have a valid value, the server MAY<115> fail the call.

If the *Flags* parameter is 0, the server MUST canonicalize the specified path names and obtain their *PathTypes* first, as specified in section 3.1.4.30. If this fails, the server MUST fail the call with ERROR\_INVALID\_NAME. If the *PathTypes* for the two path names thus obtained are different, the server MUST return 1.

The server then compares the canonicalized path names by using an implementation-specific<116> comparison and MUST return 0 to the caller if the paths match, -1 if *PathName1* is less than *PathName2*, and 1 if *PathName1* is greater than *PathName2*.

The server MAY<117> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<118> fail the call.

### 3.1.4.32 NetprNameValidate (Opnum 33)

Article 02/14/2019

The NetprNameValidate method performs checks to ensure that the specified name is a valid name for the specified type.

```
NET_API_STATUS NetprNameValidate(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* Name,
    [in] DWORD NameType,
    [in] DWORD Flags
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Name:** A pointer to a null-terminated UTF-16 string that specifies the name to check.

**NameType:** The type of *Name*. It MUST be one of the values defined in section [2.2.2.8](#).

**Flags:** Reserved, MUST be zero.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#).

If the Flags parameter is not equal to zero, the server SHOULD fail the call with an implementation-specific error code.<[119](#)>

In response to a NetprNameValidate message, the server MUST validate the value of the *Name* parameter to ensure that it contains only the characters that are allowed for the specified NameType and that the length of the *Name* parameter is no greater than the maximum allowed length for its NameType (as specified in section 2.2.2.8).

The *NameType* parameter determines what validation is done on the name that is specified by the *Name* parameter. Valid values for the *NameType* parameter are as specified in section 2.2.2.8. If the *NameType* parameter does not have a valid value, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

The value of *NameType* identifies the minimum and maximum lengths for a particular *NameType* and the characters that are permitted in a name for that *NameType*. The server MUST validate the specified name by being sure that its length is within the

minimum and maximum lengths for its type and that there are no characters in its name that are invalid for its type. If any of these checks fail, the server MUST fail the call with an ERROR\_INVALID\_NAME error code.

The server MAY [120](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD [121](#) fail the call.

## 3.1.4.33 NetprNameCanonicalize (Opnum 34)

Article 06/24/2021

The NetprNameCanonicalize method converts a name to the canonical format for the specified type.

```
NET_API_STATUS NetprNameCanonicalize(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* Name,
    [out, size_is(OutbufLen)] WCHAR* Outbuf,
    [in, range(0,64000)] DWORD OutbufLen,
    [in] DWORD NameType,
    [in] DWORD Flags
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an [RPC](#) binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Name:** A pointer to a null-terminated UTF-16 string specifying the name to canonicalize.

**Outbuf:** A pointer to a null-terminated UTF-16 string that is the buffer where the canonicalized name is returned.

**OutbufLen:** The length of output buffer *Outbuf*. The value of this field MUST be within the range 0 through 64,000, inclusive.

**NameType:** The type of *Name*, as specified in section [2.2.2.8](#).

**Flags:** A bitmask that MUST contain the bitwise OR of zero or more of the following values that specify controlling flags.

Value	Meaning
0x80000000	LM2.x compatible name canonicalization.
0x00000001	If set, the method requires the length of the output buffer to be sufficient to hold any name of the specified type. Otherwise, the buffer length only needs to be large enough to hold the canonicalized version of the input name that is specified in this invocation of the method.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code

value, as specified in [MS-ERREF] section 2.2.

In response to a NetprNameCanonicalize message, the server MUST convert the value of the *Name* parameter to one of the canonical forms that are defined in section 2.2.2.8.

The *NameType* parameter determines what needs to be done on the name that is specified by the *Name* parameter to convert it to a canonical format. Valid values for the *NameType* parameter are as specified in Name Types (section 2.2.2.8). If the *NameType* parameter does not have a valid value, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

The *Flags* parameter is a bitmask that specifies certain controlling flags that affect how the server processes this message. The valid bits are 0x80000000 and 0x1. If any other bit is set, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

If (*Flags* & 0x80000000) is true, it implies that the server MUST perform an NTLM version 2.x-compatible canonicalization. As the following table specifies, some *NameTypes* have different rules about how to define a canonical name for those types on NTLM version 2.x.

The server MUST validate the *Name* (as specified by the [NetprNameValidate](#) method) to ensure that it is a valid name of type *NameType*. If validation fails, the server MUST fail the call with ERROR\_INVALID\_NAME.

The server MUST use the *NameType* parameter to determine the maximum length of any name for that type (as specified in the following table). If (*Flags* & 0x1) is true and the length of the output buffer specified by the *OutBufLen* parameter is not greater than or equal to the maximum length of any name for that type, the server MUST fail the call with an NERR\_BufTooSmall error code.

The canonicalization process then truncates the *Name* so that the length is no greater than the maximum length for that type, converting the name to uppercase if needed. The following table specifies the maximum length for each *NameType* and whether the server converts names to uppercase. The second column in the table specifies the behavior when (*Flags* & 0x80000000) is true, and the third column specifies the behavior when it is false.

NameType	Max name length for NTLM 2.x mode / Uppercase	Max name length otherwise / Uppercase
NAMETYPE_USER	20/YES	256/NO
1		

NameType	Max name length for NTLM 2.x mode / Uppercase	Max name length otherwise / Uppercase
NAMETYPE_PASSWORD 2	14/NO	256/NO
NAMETYPE_GROUP 3	20/YES	256/NO
NAMETYPE_COMPUTER 4	15/YES	259/NO
NAMETYPE_EVENT 5	16/YES	16/YES
NAMETYPE_DOMAIN 6	15/YES	15/NO
NAMETYPE_SERVICE 7	15/YES	80/NO
NAMETYPE_NET 8	259/YES	259/YES
NAMETYPE_SHARE 9	12/YES	80/NO
NAMETYPE_MESSAGE 10	259/YES	259/YES
NAMETYPE_MESSAGEDEST 11	259/YES	259/YES
NAMETYPE_SHAREPASSWORD 12	8/NO	8/NO
NAMETYPE_WORKGROUP 13	15/YES	15/NO

The server MAY [<122>](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures

and the caller does not have the required credentials, the server SHOULD [123](#) fail the call.

## 3.1.4.34 NetprNameCompare (Opnum 35)

Article 06/24/2021

The NetprNameCompare method does comparison of two names of a specific name type.

```
long NetprNameCompare(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* Name1,
    [in, string] WCHAR* Name2,
    [in] DWORD NameType,
    [in] DWORD Flags
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the **server**. The **client** MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Name1:** A pointer to a null-terminated UTF-16 string that contains the first name to compare.

**Name2:** A pointer to a null-terminated UTF-16 string that contains the second name to compare.

**NameType:** The type of names, as specified in section [2.2.2.8](#).

**Flags:** A bitmask that MUST contain the bitwise OR of zero or more of the following values, which specify controlling flags.

Value	Meaning
0x80000000	Enable LM2.x compatibility.
0x00000001	SHOULD be set if both names have already been canonicalized (by using NetprNameCanonicalize).

**Return Values:** MUST return 0 if both paths are the same. Other values indicate that either the paths are different or an error occurred when the client request was processed.

In response to a NetprNameCompare message, the server MUST compare the two names that are specified as parameters to ensure that they contain only the characters

that are allowed for the specified NameType and that the length is no greater than the maximum allowed length for its NameType (as specified in section 2.2.2.8). If the supplied names are not canonicalized, the server MUST do the canonicalization of the names.

The *Name1* parameter and *Name2* parameter specify the two names to be compared.

The *Flags* parameter is a bitmask that specifies certain controlling flags that affect how the server processes this message. The valid bits are 0x80000000 and 0x1. If any other bit is set, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

If (*Flags* & 0x80000000) is true, it implies that the server MUST enable NTLM version 2.x compatibility. This implies that the rules that are used for comparison and canonicalization (if needed) MUST be those that are defined for NTLM version 2.x. For details about the effect on canonicalization, see NetprNameCanonicalize (Opnum 34) (section 3.1.4.33). With respect to comparison, if (*Flags* & 0x80000000) is true and the NameType being compared is NAMETYPE\_PASSWORD, NAMETYPE\_SHAREPASSWORD, NAMETYPE\_MESSAGE, or NAMETYPE\_MESSAGEDEST, the server MUST perform a case-sensitive comparison. Otherwise, the server MUST perform a case-insensitive comparison.

If (*Flags* & 0x1) is true, the names that are specified by *Name1* and *Name2* are already canonicalized, and the *NameType* parameter MUST specify the name type for the two names. Valid values for the *NameType* parameter are listed in Name Types (section 2.2.2.8). If the *NameType* parameter does not have a valid value, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

If (*Flags* & 0x1) is not true, the server MUST canonicalize the specified names and obtain their name types, as specified in NetprNameCanonicalize (section 3.1.4.33). If this fails, the server MUST fail the call with an ERROR\_INVALID\_PARAMETER error code.

The server MUST compare the canonicalized version of the names, if the names were not already canonicalized; otherwise, it MUST compare the original names and MUST return 0 if both names are the same, -1 if *Name1* is less than *Name2*, and 1 if *Name1* is greater than *Name2*. The comparison is implementation-specific.<124>

The server MAY<125> enforce security measures to verify that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD<126> fail the call.

## 3.1.4.35 NetrDfsGetVersion (Opnum 43)

Article06/24/2021

The NetrDfsGetVersion method checks whether the [server](#) is a [DFS](#) server and if so, returns the DFS version. An implementation MAY[<sup>127</sup>](#) choose to support this method.

```
NET_API_STATUS NetrDfsGetVersion(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [out] DWORD* Version
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) ([section 2.2.1.1](#)) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) [<sup>128</sup>](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Version:** A pointer to a DWORD where the server returns the DFS version.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#).

In response to a NetrDfsGetVersion message, the server SHOULD[<sup>128</sup>](#) choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports DFS, the server MAY return the DFS version number that is in use on the server.

The *Version* parameter is a pointer to a DWORD. If the server supports DFS, the server MUST set this parameter to an implementation-specific[<sup>129</sup>](#) DFS version number that the server supports.

The server MAY[<sup>130</sup>](#) enforce security measures to verify that the server enforces these security measures and that the caller has the required permissions to execute this call. If the caller does not have the required credentials, the server SHOULD[<sup>131</sup>](#) fail the call.

## 3.1.4.36 NetrDfsCreateLocalPartition (Opnum 44)

Article 06/24/2021

The `NetrDfsCreateLocalPartition` method marks a share as being a [DFS](#) share. In addition, if the `RelationInfo` parameter is non-NULL, it creates [DFS links](#), as specified in [\[MS-DFSC\]](#), for each of the entries in the `RelationInfo` parameter. An implementation MAY<132> choose to support this method.

```
NET_API_STATUS NetrDfsCreateLocalPartition(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* ShareName,
    [in] GUID* EntryUid,
    [in, string] WCHAR* EntryPrefix,
    [in, string] WCHAR* ShortName,
    [in] LPNET_DFS_ENTRY_ID_CONTAINER RelationInfo,
    [in] int Force
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**ShareName:** A pointer to a null-terminated UTF-16 string that specifies the name of a local disk share on the server to add to DFS.

**EntryUid:** A pointer to a [GUID](#) type that specifies the GUID for this DFS share. The GUID for this share MUST NOT match a GUID for an existing local partition.<133>

**EntryPrefix:** A pointer to a null-terminated UTF-16 string that specifies the path of the DFS share.

**ShortName:** A pointer to a null-terminated UTF-16 string that specifies the short-name version (8.3 format) of the `EntryPrefix` parameter.

**RelationInfo:** A pointer to a [NET\\_DFS\\_ENTRY\\_ID\\_CONTAINER](#) structure. Specifies the DFS child links that are under the DFS share that is specified by the `EntryPrefix` parameter.

**Force:** The `Force` parameter is ignored and MUST be set to zero.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a `NetrDfsCreateLocalPartition` message, the server SHOULD <134> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports DFS, the server MAY mark an existing SMB file share as a DFS share that enables it to be accessed by using DFS, as specified in [MS-DFSC].

The *ShareName* parameter MUST specify the name of an existing SMB file share of type STYPE\_DISKTREE (for more information, see [Share Types \(section 2.2.2.4\)](#)), or the server MUST fail the call with an `ERROR_BAD_NET_NAME` error code if the share is not present. If the share is present, but not of type STYPE\_DISKTREE, it MUST fail with an `ERROR_BAD_DEV_TYPE` error code.

The *EntryUid* parameter specifies the GUID that the server MUST assign to the new DFS share.

This parameter MUST NOT be NULL, or the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code. If the *EntryUid* parameter matches a GUID for an existing local partition, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The *EntryPrefix* parameter specifies the path of the DFS share. This string MUST be in one of the following two forms:

- The first form is `\Dfsname\sharename`, where *Dfsname* is the name of the storage server that hosts the root of a [standalone DFS implementation](#); and *sharename* is the name of a shared folder that is published on the DFS host server.
- The second form is `\DomainName\DomDfsname`, where *DomainName* is the name of the domain that hosts the [DFS root](#); and *DomDfsname* is the name of the root of a domain-based DFS implementation that is published in the directory service of the domain.

The *RelationInfo* parameter specifies the DFS child links to create under the share that is specified by *EntryPrefix*. It has a member count that specifies the number of child links and a Buffer member that is an array of the Count structure of type [NET\\_DFS\\_ENTRY\\_ID](#). A DFS child link MUST be created for each entry in the Buffer. The *RelationInfo* parameter MUST not be NULL, or the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

The *ShortName* parameter specifies a share name (in the 8.3 format) that is specified by *EntryPrefix* and MUST be interpreted by the server in an implementation-specific manner. <135>

The *Force* parameter is ignored and MUST be zero.

The server MAY<136> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<137> fail the call.

## 3.1.4.37 NetrDfsDeleteLocalPartition (Opnum 45)

Article 06/24/2021

The NetrDfsDeleteLocalPartition method deletes a [DFS](#) share (Prefix) on the [server](#). An implementation MAY <138> choose to support this method.

```
NET_API_STATUS NetrDfsDeleteLocalPartition(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] GUID* Uid,
    [in, string] WCHAR* Prefix
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Uid:** Specifies the [GUID](#) of the DFS share to delete. This GUID MUST be obtained by using the NetrDfsGetInfo (Opnum 4) method, which is specified in [\[MS-DFSNM\]](#) section 3.1.4.1.6.

**Prefix:** A pointer to a null-terminated UTF-16 string that contains the path to the DFS share.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrDfsDeleteLocalPartition message, the server SHOULD <139> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports DFS, the server MAY delete a DFS share.

The *Prefix* parameter specifies the path of the DFS share to delete. This string MUST be in one of the following two forms:

- The first form is \Dfsname\sharename, where *Dfsname* is the name of the storage server that hosts the root of a [standalone DFS implementation](#); and *sharename* is the name of a shared folder that is published on the DFS host server.
- The second form is \DomainName\DomDfsname, where *DomainName* is the name of the domain that hosts the [DFS root](#); and *DomDfsname* is the root name of a

domain-based DFS implementation that is published in the directory service of the domain.

If the server cannot find a DFS share whose GUID matches the *Uid* parameter and whose path matches the *Prefix* parameter, it MUST fail the call with an implementation-specific error code. If a matching share is found, the server deletes the share and returns 0.

The server MAY<sup><140></sup> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<sup><141></sup> fail the call.

## 3.1.4.38 NetrDfsSetLocalVolumeState (Opnum 46)

Article 06/24/2021

The NetrDfsSetLocalVolumeState method sets a local DFS share online or offline. An implementation MAY <142> choose to support this method.

```
NET_API_STATUS NetrDfsSetLocalVolumeState(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] GUID* Uid,
    [in, string] WCHAR* Prefix,
    [in] unsigned long State
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Uid:** Specifies the [GUID](#) of the DFS share. This GUID MUST be obtained by using the NetrDfsGetInfo (Opnum 4) method, as specified in [\[MS-DFSNM\]](#) section 3.1.4.1.6.

**Prefix:** A pointer to a null-terminated UTF-16 string that specifies the path to the DFS share.

**State:** A DWORD that specifies the new state for the DFS share. To set the share to offline, the *State* parameter MUST be (0x80). The *State* parameter MUST be set to any other value to take the share online.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrDfsSetLocalVolumeState message, the server SHOULD <143> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports DFS, the server MAY set the state of a local DFS share to online or offline. Marking a share state offline makes the share inaccessible over DFS.

The *Uid* parameter specifies the GUID of the share whose state needs to be set.

The *Prefix* parameter specifies the path of the DFS share whose state needs to be set. This parameter MUST refer to a local DFS share. If the server does not find a DFS share

whose path starts with the value of the *Prefix* parameter and whose GUID matches the value of the *Uid* parameter, the server MUST fail the call and return an implementation-specific error code.

The *State* parameter specifies whether the share state MUST be set to online or offline. If the value of *State* is 0x80, the share state MUST be set to offline. For any other value, the share state MUST be set to online.

The server MAY<sup><144></sup> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<sup><145></sup> fail the call.

## 3.1.4.39 NetrDfsCreateExitPoint (Opnum 48)

Article 02/14/2019

The NetrDfsCreateExitPoint method creates a [DFS link](#) on the [server](#). An implementation MAY<146> choose to support this method.

```
NET_API_STATUS NetrDfsCreateExitPoint(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] GUID* Uid,
    [in, string] WCHAR* Prefix,
    [in] unsigned long Type,
    [in, range(0,32)] DWORD ShortPrefixLen,
    [out, size_is(ShortPrefixLen)] WCHAR* ShortPrefix
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) ↗ sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Uid:** Specifies the [GUID](#) for the DFS link. This GUID MUST be obtained by using the NetrDfsGetInfo (Opnum 4) method, which is specified in [\[MS-DFSNM\]](#) section [3.1.4.1.6](#).

**Prefix:** A pointer to a null-terminated UTF-16 string that specifies the path of the DFS link.

**Type:** This parameter MUST be one of the values that are specified in section [2.2.2.13](#).

**ShortPrefixLen:** Specifies the size of the buffer passed in the *ShortPrefix*. The value of this field MUST be within the range 0 through 32, inclusive.

**ShortPrefix:** A pointer to a null-terminated UTF-16 string that is the buffer where the name of the DFS namespace root or link is returned.<147>

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section [2.2](#).

In response to a NetrDfsCreateExitPoint message, the server SHOULD<148> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports [DFS](#), the server MAY create a DFS link, as specified in [\[MS-DFSC\]](#).

The *Uid* parameter specifies the GUID to be assigned to the new link.

The *Prefix* parameter specifies the path of the DFS link. The string MUST be in one of two forms:

- The first form is \Dfsname\sharename\path\_to\_link, where *Dfsname* is the name of the storage server that hosts the root of a [standalone DFS implementation](#); *sharename* is the name of a shared folder that is published on the DFS host server; and *path\_to\_link* specifies the path on the physical network share.
- The second form is \DomainName\DomDfsname\path\_to\_link, where *DomainName* is the name of the domain that hosts the [DFS root](#); *DomDfsname* is the root name of a domain-based DFS implementation that is published in the directory service of the domain; and *path\_to\_link* specifies the path on the physical network share.

The *Type* parameter specifies the type of the new link and MUST be one of the values listed in section 2.2.2.13. If the value of this parameter is PKT\_ENTRY\_TYPE\_MACHINE, the server MUST fail the call and return an implementation-specific error code.

The *ShortPrefixLen* parameter specifies the length of the *ShortPrefix* parameter that specifies a short name for the new link in the 8.3 format.

The server MAY [<149>](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD [<150>](#) fail the call.

## 3.1.4.40 NetrDfsModifyPrefix (Opnum 50)

Article 06/24/2021

The NetrDfsModifyPrefix method changes the path that corresponds to a [DFS link](#) on the [server](#). An implementation MAY<151> choose to support this method.

```
NET_API_STATUS NetrDfsModifyPrefix(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] GUID* Uid,
    [in, string] WCHAR* Prefix
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Uid:** Specifies the [GUID](#) that corresponds to the DFS link that needs to be changed. This GUID MUST be obtained by using the NetrDfsGetInfo (Opnum 4) method, specified in [\[MS-DFSNM\]](#) section 3.1.4.1.6.

**Prefix:** A pointer to a null-terminated UTF-16 string that specifies the path of the updated DFS link.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrDfsModifyPrefix message, the server SHOULD<152> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports [DFS](#), the server MAY update the path for a DFS link. This message is typically used by domain controllers (DCs) to fix a bad prefix match.

The *Uid* parameter specifies the GUID that corresponds to the DFS link that needs to be changed.

The *Prefix* parameter specifies the path of the updated DFS link. The string MUST be in one of two forms:

- The first form is \Dfsname\sharename\path\_to\_link, where *Dfsname* is the name of the storage server that hosts the root of a [standalone DFS implementation](#);

*sharename* is the name of a shared folder that is published on the DFS host server; and *path\_to\_link* specifies the path on the physical network share.

- The second form is \DomainName\DomDfsname\path\_to\_link, where *DomainName* is the name of the domain that hosts the [DFS root](#); *DomDfsname* is the name of the root of a domain-based DFS implementation that is published in the directory service of the domain; and *path\_to\_link* specifies the path on the physical network share.

The server MAY [<153>](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD [<154>](#) fail the call.

## 3.1.4.41 NetrDfsDeleteExitPoint (Opnum 49)

Article 06/24/2021

The NetrDfsDeleteExitPoint method deletes a [DFS link](#) on the [server](#). An implementation MAY<155> choose to support this method.

```
NET_API_STATUS NetrDfsDeleteExitPoint(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] GUID* Uid,
    [in, string] WCHAR* Prefix,
    [in] unsigned long Type
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) point that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Uid:** Specifies the [GUID](#) that corresponds to the DFS link that is specified by the *Prefix* parameter. This GUID MUST be obtained by using the NetrDfsGetInfo (Opnum 4) method, specified in [\[MS-DFSNM\]](#) section 3.1.4.1.6.

**Prefix:** A pointer to a null-terminated UTF-16 string that specifies the path of the DFS link.

**Type:** This parameter MUST be one of the values listed in section 2.2.2.13.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

In response to a NetrDfsDeleteExitPoint message, the server SHOULD<156> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports DFS, the server MAY delete a DFS link, as specified in [\[MS-DFSC\]](#).

The *Uid* parameter specifies the GUID of the link to delete.

The *Prefix* parameter specifies the path of the DFS link. The string MUST be in one of two forms:

- The first form is \Dfsname\sharename\path\_to\_link, where *Dfsname* is the name of the storage server that hosts the root of a [standalone DFS implementation](#);

*sharename* is the name of a shared folder that is published on the [DFS](#) host server; and *path\_to\_link* specifies the path on the physical network share.

- The second form is \DomainName\DomDfsname\path\_to\_link, where *DomainName* is the name of the domain that hosts the [DFS root](#); *DomDfsname* is the root name of a domain-based DFS implementation that is published in the directory service of the domain; and *path\_to\_link* specifies the path on the physical network share.

The *Type* parameter specifies the type of the link to delete and MUST be one of the values listed in section 2.2.2.13. If the value of this parameter is `PKT_ENTRY_TYPE_MACHINE`, the server MUST fail the call and return an implementation-specific error code.

If a link whose GUID, path, and type match the specified parameters is present, the server MUST delete it; otherwise, it MUST fail the call with an implementation-specific error code.

The server MAY [<157>](#) enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD [<158>](#) fail the call.

## 3.1.4.42 NetrDfsFixLocalVolume (Opnum 51)

Article 06/24/2021

The NetrDfsFixLocalVolume method provides knowledge of a new DFS share on the server. An implementation MAY<159> choose to support this method.

```
NET_API_STATUS NetrDfsFixLocalVolume(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, string] WCHAR* VolumeName,
    [in] unsigned long EntryType,
    [in] unsigned long ServiceType,
    [in, string] WCHAR* StgId,
    [in] GUID* EntryUid,
    [in, string] WCHAR* EntryPrefix,
    [in] LPNET_DFS_ENTRY_ID_CONTAINER RelationInfo,
    [in] unsigned long CreateDisposition
);
```

**ServerName:** An [SRVSVC\\_HANDLE \(section 2.2.1.1\)](#) pointer that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**VolumeName:** A pointer to a null-terminated UTF-16 string that specifies the target for the [DFS root](#) share. This target MUST be local to the server; for example, \??\C:\DfsShare.<160> This target SHOULD NOT contain a directory that is in DFS, and it SHOULD NOT be a child of a DFS share. If the specified volume name is not valid, the server SHOULD fail the call by using an implementation-specific error code.

**EntryType:** This parameter MUST be one of the values listed in section [2.2.2.13](#). If the specified entry type is not valid, the server SHOULD fail the call with an implementation-specific error code.

**ServiceType:** This parameter MUST be a combination of one or more of the following values. If the specified service type is not valid, the server SHOULD fail the call with an implementation-specific error code.

Value	Meaning
DFS_SERVICE_TYPE_MASTER	Master service
0x00000001	

Value	Meaning
DFS_SERVICE_TYPE_READONLY 0x00000002	Read-only service
DFS_SERVICE_TYPE_LOCAL 0x00000004	Local service
DFS_SERVICE_TYPE_REFERRAL 0x00000008	Referral service
DFS_SERVICE_TYPE_ACTIVE 0x00000010	Active service
DFS_SERVICE_TYPE_DOWN_LEVEL 0x00000020	Down-level service
DFS_SERVICE_TYPE_COSTLIER 0x00000040	Costlier service than previous
DFS_SERVICE_TYPE_OFFLINE 0x00000080	Service is offline

**StgId:** A pointer to a variable that specifies an ID for the local storage. The server MUST ignore the value that is passed in for the *StgId* parameter.

**EntryUid:** Specifies the [GUID](#) that corresponds to the DFS share. This GUID MUST be obtained by using the NetrDfsGetInfo (Opnum 4) method, which is specified in [\[MS-DFSNM\]](#) section [3.1.4.1.6](#).

**EntryPrefix:** A pointer to a null-terminated UTF-16 string that specifies the path of the DFS share to be updated.

**RelationInfo:** A pointer to a [NET\\_DFS\\_ENTRY\\_ID\\_CONTAINER](#) structure as specified in section 2.2.4.108. Specifies the [DFS child links](#) under the DFS share as specified by the *EntryPrefix* parameter.

**CreateDisposition:** Specifies what to do, depending on whether the share already exists. This field MUST be set to one of the following values.

Value	Meaning
-------	---------

Value	Meaning
FILE_SUPERSEDE 0x00000000	If the share already exists, replace it with the specified share. If it does not exist, create the specified share.
FILE_OPEN 0x00000001	If the share already exists, fail the request and do not create or open the specified share. If it does not exist, create the specified share.
FILE_CREATE 0x00000002	If the file already exists, open it instead of creating a new share. If it does not exist, fail the request and do not create a new share.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2.

In response to a NetrDfsFixLocalVolume message, the server SHOULD<161> choose to perform no processing and return an implementation-specific error code when this method is called. If the server supports DFS, the server MAY add the link name that corresponds to a specified Uid. This message typically is sent by a domain controller when it discovers that the server is completely unaware of a new DFS volume.

The *VolumeName* parameter specifies the target for the DFS root share. This target MUST be local to the server and is in the form of a Windows NT operating system path name, for example, \?\?C:\DfsShare.<162> This target SHOULD NOT contain a directory that is in DFS, and it SHOULD NOT be a child of a DFS share.

The *EntryType* parameter specifies the type of the link and MUST be one of the values listed in section 2.2.2.13.

The *ServiceType* parameter specifies the service type of the client.

The *StgId* parameter specifies an implementation-specific ID for the local storage.

The *EntryUid* parameter specifies the GUID for the new link.

The *Prefix* parameter specifies the path of the updated DFS link. The string MUST be in one of two forms:

- The first form is \Dfsname\sharename\path\_to\_link, where *Dfsname* is the name of the storage server that hosts the root of a [standalone DFS implementation](#); *sharename* is the name of a shared folder that is published on the DFS host server; and *path\_to\_link* specifies the path on the physical network share.

- The second form is \DomainName\DomDfsname\path\_to\_link, where *DomainName* is the name of the domain that hosts the DFS root; *DomDfsname* is the name of the root of a domain-based DFS implementation that is published in the directory service of the domain; and *path\_to\_link* specifies the path on the physical network share.

The *RelationInfo* parameter specifies the DFS child links under the DFS share that is specified by *EntryPrefix*. If this parameter is NULL or if its **Count** member is nonzero and its **Buffer** member is NULL, the server fails the call by using an ERROR\_INVALID\_PARAMETER error code.

The *CreateDisposition* parameter specifies what MUST happen if a share with the path *EntryPrefix* already exists.

The server MAY<sup><163></sup> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<sup><164></sup> fail the call.

## 3.1.4.43 NetrDfsManagerReportSiteInfo (Opnum 52)

Article04/06/2021

The NetrDfsManagerReportSiteInfo method obtains a list of names that SHOULD <165> correspond to the Active Directory [sites](#) covered by the specified server. An implementation MAY <166> choose to support this method.

```
NET_API_STATUS NetrDfsManagerReportSiteInfo(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, out, unique] LPDFS_SITELIST_INFO* ppSiteInfo
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the [server](#). The [client](#) MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2).

**ppSiteInfo:** A pointer to an [LPDFS\\_SITELIST\\_INFO](#) structure, which in turn points to the location of a [DFS\\_SITELIST\\_INFO](#) structure in which the information is returned.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2.

The *ppSiteInfo* parameter is a pointer to a [LPDFS\\_SITELIST\\_INFO](#) member, which in turn points to the location of a [DFS\\_SITELIST\\_INFO](#) structure in which the information is returned. That structure has a **cSites** member that the server SHOULD set to the number of sites returned. The information about the sites themselves MUST be returned in the **Site** member, which is an array of [DFS\\_SITENAME\\_INFO](#) structures. The sites the server returns are implementation-specific.<167>

The server MAY <168> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD <169> fail the call.

## 3.1.4.44 NetrServerAliasAdd (Opnum 54)

Article 06/24/2021

The NetrServerAliasAdd method attaches an alias name to an existing [server](#) name and inserts Alias objects into [AliasList](#), through which the shared resource can be accessed either with server name or alias name. An alias is used to identify which resources are visible to an [SMB](#) client based on the server name presented in each tree connect request.

```
NET_API_STATUS NetrServerAliasAdd(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSEVER_ALIAS_INFO InfoStruct
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. It MUST be one of the following values.

Value	Meaning
0	The buffer is of type <a href="#">SERVER_ALIAS_INFO_0_CONTAINER</a> .

**InfoStruct:** A pointer to the [SERVER\\_ALIAS\\_INFO](#) union that contains information about the alias. The value of the *Level* parameter determines the type of the contents of the *InfoStruct* parameter, as the preceding table shows.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000	The <a href="#">client</a> request succeeded.
NERR_Success	

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000846 NERR_DuplicateShare	The alias already exists.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.

In response to a `NetrServerAliasAdd` message, the server MUST add an alias to attach the existing server name and insert it into **AliasList** upon successful return, or return an error code for a failure case. Multiple alias names can be attached to the same server name.

The server name to be attached to the alias is specified in the `srvai*_target` member of the `SERVER_ALIAS_INFO` structure. If the specified target name is an empty string or does not match any `Transport.ServerName` in the `TransportList`, the server SHOULD fail the call with an `ERROR_INVALID_PARAMETER` error code.

The *Level* parameter determines the type of structure that the client has used to specify information about the new alias. The value of the *Level* parameter MUST be 0. If the *Level* parameter is not equal to 0, the server MUST fail the call and return an `ERROR_INVALID_LEVEL` error code.

The name of the alias to be added is specified in the `srvai*_alias` member of the `SERVER_ALIAS_INFO` structure. `srvai*_alias` MUST be a nonempty null-terminated UTF-16 string if `srvai*_default` is 0 or an empty string if `srvai*_default` is nonzero; otherwise, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code. If `srvai*_alias` is a nonempty string and it matches an existing `Alias.alias` in the `AliasList`, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code. If `srvai*_alias` is an empty string and `srvai*_default` is set, the server MUST fail the call with an implementation-specific error code if `DefaultServerName` is not NULL. Otherwise, `DefaultServerName` MUST be set to `srvai*_target` as specified in section 3.1.1.1.

The server MAY<170> enforce security measures to verify that the caller has the required permissions to execute this call. If the server enforces these security measures and the caller does not have the required credentials, the server SHOULD<171> fail the call.

The server MUST persist the *InfoStruct* and *Level* parameters to a persistent configuration store. If an alias with the same *srvai0\_alias* and *srvai0\_target* already exists in the store, the preexisting entry MUST be overwritten with this entry.

## 3.1.4.45 NetrServerAliasEnum (Opnum 55)

Article 06/24/2021

The NetrServerAliasEnum method retrieves alias information for a [server](#) based on specified alias name or server name.

```
NET_API_STATUS NetrServerAliasEnum(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in, out] LP SERVER_ALIAS_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] LPDWORD TotalEntries,
    [in, out, unique] LPDWORD ResumeHandle
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The client MUST map this structure to an RPC binding handle (see [\[C706\]](#) sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**InfoStruct:** A pointer to a structure, in the format of a [SERVER\\_ALIAS\\_ENUM\\_STRUCT](#), as specified in section 2.2.4.104. The SERVER\_ALIAS\_ENUM\_STRUCT structure has a **Level** member that specifies the type of structure to return in the **ServerAliasInfo** member. The **Level** member MUST be one of the values specified in section 2.2.4.104.

**PreferredMaximumLength:** Specifies the preferred maximum length, in bytes, of the returned data. If the specified value is [MAX\\_PREFERRED\\_LENGTH](#), the method MUST attempt to return all entries.

**TotalEntries:** The total number of entries that could have been enumerated if the buffer had been big enough to hold all the entries.

**ResumeHandle:** A pointer to a value that contains a handle, which is used to continue an existing alias search in **AliasList**. The handle MUST be zero on the first call and remain unchanged for subsequent calls. If the *ResumeHandle* parameter is NULL, no resume handle MUST be stored. If this parameter is not NULL and the method returns **ERROR\_MORE\_DATA**, this parameter receives an implementation-specific nonzero value that can be passed in subsequent calls to this method to continue with the enumeration.

If this parameter is NULL or points to 0x00000000, the enumeration starts from the beginning of the **AliasList**.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [MS-ERREF] section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The <a href="#">client</a> request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000084B NERR_BufTooSmall	The allocated buffer is too small to hold single entry.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x000000EA ERROR_MORE_DATA	The client request succeeded. More entries are available. Not all entries could be returned in the buffer size that is specified by <i>PreferredMaximumLength</i> .

In response to a NetrServerAliasEnum message, the server MUST return information about each alias resource on a server, or return an error code.

The *InfoStruct* parameter has a **Level** member. The valid values of **Level** are 0. If the **Level** member is not equal to 0, the server MUST fail the call with an **ERROR\_INVALID\_LEVEL** error code.

If the **Level** member is 0, the server MUST return the information about aliases by filling the **SERVER\_ALIAS\_INFO\_0\_CONTAINER** structure in the **ServerAliasInfo** member of the *InfoStruct* parameter. The **SERVER\_ALIAS\_INFO\_0\_CONTAINER** structure contains an array of **SERVER\_ALIAS\_INFO\_0** structures.

The *PreferredMaximumLength* parameter specifies the maximum number of bytes that the server can return for the **ServerAliasInfo** buffer. If *PreferredMaximumLength* is

insufficient to hold all the entries, the server MUST return the maximum number of entries as will fit in the **ServerAliasInfo** buffer and return **ERROR\_MORE\_DATA**. If this parameter is equal to **MAX\_PREFERRED\_LENGTH**, the server MUST return all the requested data.

If the server returns **NERR\_Success** or **ERROR\_MORE\_DATA**, it MUST set the *TotalEntries* parameter to equal the total number of entries that could have been enumerated from the current resume position.

If *PreferredMaximumLength* is insufficient to hold all the entries and if the client has specified a *ResumeHandle*, the server MUST set *ResumeHandle* to some implementation-specific value that allows the server to continue with this enumeration on a subsequent call to this method with the same value for *ResumeHandle*.

The server MUST maintain **AliasList**.

The following rules specify processing of the *ResumeHandle* parameter:

- If the *ResumeHandle* parameter is either **NULL** or points to **0x00000000**, the enumeration MUST start from the beginning of the list of the **AliasList**.
- If the *ResumeHandle* parameter points to a nonzero value, the server MUST continue enumeration based on the value of *ResumeHandle*. The value of *ResumeHandle* specifies the index into the **AliasList** after which the enumeration is to begin.
- If the client specified a *ResumeHandle* and if the server returns **ERROR\_MORE\_DATA** (**0x000000EA**), the server MUST set *ResumeHandle* to the index of the last enumerated alias in the **AliasList**.

Because the *ResumeHandle* specifies an offset into the list, and the list of aliases can be modified between multiple requests, the results of a query spanning multiple requests using the *ResumeHandle* can be unreliable, offering either duplicate or missed aliases.

The server SHOULD<[172](#)> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<[173](#)> fail the call.

## 3.1.4.46 NetrServerAliasDel (Opnum 56)

Article10/30/2020

The NetrServerAliasDel method deletes an alias name from a [server](#) alias list based on specified alias name.

```
NET_API_STATUS NetrServerAliasDel(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSERVER_ALIAS_INFO InfoStruct
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) (section 2.2.1.1) pointer that identifies the server. The client MUST map this structure to an RPC binding handle ([\[C706\]](#) sections 4.3.5 and 5.1.5.2). If this parameter is NULL, the local computer is used.

**Level:** Specifies the information level of the data. It MUST be one of the following values.

Value	Meaning
0	The buffer is of type <a href="#">SERVER_ALIAS_INFO_0_CONTAINER</a> .

**InfoStruct:** A pointer to the [SERVER\\_ALIAS\\_INFO](#) union that contains information about the alias. The value of the *Level* parameter determines the type of the contents of the *InfoStruct* parameter, as the preceding table shows.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000 NERR_Success	The <a href="#">client</a> request succeeded.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.

Return value/code	Description
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000906 NERR_NetNameNotFound	The alias does not exist.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.

In response to a `NetrServerAliasDel` message, the server MUST delete the alias name from the **AliasList** based on specified alias name, or MUST return an error code.

The *srvali\*\_alias* parameter specifies the name of the alias to be deleted. This MUST be a nonempty null-terminated UTF-16 string if *srvali\*\_default* is 0 or empty string if *srvali\*\_default* is nonzero; otherwise, the server MUST fail the call with an `ERROR_INVALID_PARAMETER` error code.

If no alias matching *srvali\*\_alias* exists, the server fails the call with a `NERR_NetNameNotFound` error code.

*srvali\*\_target* MUST be ignored by the server.

The server SHOULD [174](#) enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD [175](#) fail the call.

The server MUST delete configuration data for this alias from the persistent configuration store.

## 3.1.4.47 NetrShareDelEx (Opnum 57)

Article 10/30/2020

The NetrShareDelEx method deletes a share from the **ShareList**, which disconnects all connections to the shared resource. If the share is sticky, all information about the share is also deleted from permanent storage.<176>

```
NET_API_STATUS NetrShareDelEx(
    [in, string, unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSHARE_INFO ShareInfo
);
```

**ServerName:** An [SRVSVC\\_HANDLE](#) pointer that identifies the [server](#). The client MUST map this structure to an RPC binding handle ([\[C706\]](#) ↗ sections 4.3.5 and 5.1.5.2). The server MUST ignore this parameter.

**Level:** Specifies the information level of the data. This parameter MUST be one of the following values.

Value	Meaning
503	<a href="#">LPSHARE_INFO_503_I</a>

**ShareInfo:** This parameter is of type [LPSHARE\\_INFO](#) union, as specified in section 2.2.3.6. Its contents are determined by the value of the *Level* parameter, as shown in the preceding table. This parameter MUST NOT contain a null value.

**Return Values:** The method returns 0x00000000 (NERR\_Success) to indicate success; otherwise, it returns a nonzero error code. The method can take any specific error code value, as specified in [\[MS-ERREF\]](#) section 2.2. The most common error codes are listed in the following table.

Return value/code	Description
0x00000000	The <a href="#">client</a> request succeeded.
NERR_Success	
0x00000005	Access is denied.
ERROR_ACCESS_DENIED	

Return value/code	Description
0x00000057 ERROR_INVALID_PARAMETER	The client request failed because the specified parameter is invalid.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000906 NERR_NetNameNotFound	The share name does not exist.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.

The *ShareInfo.shi503\_netname* parameter specifies the name of the share to delete from the **ShareList**. This MUST be a nonempty null-terminated UTF-16 string; otherwise, the server MUST fail the call with an **ERROR\_INVALID\_PARAMETER** error code.

The server MUST provide tuple <*ShareInfo.shi503\_servername*, *ShareInfo.shi503\_netname*> to look up the Share as specified in section 3.1.6.1. If no match is found, the server MUST fail the call with a **NERR\_NetNameNotFound** (0x00000906) error code. If the Share is found and **Share.IsPrinterShare** is TRUE, **PrinterShareCount** MUST be decreased by 1. If **PrinterShareCount** becomes 0, the server MUST invoke an event as specified in section 3.1.6.10, providing **SV\_TYPE\_PRINTQ\_SERVER** as the input parameter. The server MUST remove the share entry from **ShareList**.

In response to a **NetrShareDelEx** message, the server MUST delete the Share by invoking the underlying server event as specified in [MS-CIFS] section 3.3.4.11 and [MS-SMB2] section 3.3.4.15, providing the tuple <*ShareInfo.shi503\_servername*, *ShareInfo.shi503\_netname*> as input parameters. If the event fails, the server MUST return an error code.

The server SHOULD<177> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD<178> fail the call.

## 3.1.5 Timer Events

Article02/14/2019

No protocol timer events are required on the [client](#) beyond the timers that are required in the underlying [RPC](#) transport.

## **3.1.6 Other Local Events**

Article02/14/2019

None.

## 3.1.6.1 Server Looks Up Shares

Article04/06/2021

The server MUST provide the tuple <ServerName, ShareName> to look up shares in ShareList, as specified in section [3.1.1.1](#).

**ShareName:** The name of a shared resource. This MUST not be an empty string.

**ServerName:** The name of a local server to which the shared resource attaches. This could be an empty string.

To look up the share(s) in ShareList, the following algorithm MUST be used.

```
FOREACH Share in ShareList
    IF Share.Name is equal to ShareName
        IF Share.ServerName is equal to ServerName
            RETURN Share
        ENDIF
    ENDIF
ENDFOR
RETURN NULL
```

## 3.1.6.2 Server Registers a New Session

Article02/14/2019

The CIFS or SMB2 server requesting registration of a Session provides no parameters. The server MUST insert a new Session into **SessionList**, and MUST assign *Session.GlobalSessionId* the value that uniquely identifies the entry in the list. This value MUST be returned to the caller.

### 3.1.6.3 Server Deregisters a Session

Article02/14/2019

The CIFS or SMB2 server MUST provide the SessionId of the Session that is being deregistered.

The server MUST look up the Session in **SessionList** where Session.GlobalSessionId is equal to the SessionId provided by the caller, and remove it from **SessionList**.

## 3.1.6.4 Server Registers a New Open

Article02/14/2019

The CIFS or SMB2 server requesting registration of an Open provides no parameters. The server MUST insert a new Open into **FileList**, and MUST assign *Open.GlobalFileId* a value that uniquely identifies the entry in the list. This value MUST be returned to the caller.

## 3.1.6.5 Server Deregisters an Open

Article02/14/2019

The CIFS or SMB2 server MUST provide the *FileId* of the **Open** that is being deregistered.

The server MUST look up the **Open** in **FileList**, where **Open.GlobalFileId** is equal to the *FileId* provided by the caller, and remove it from **FileList**.

## 3.1.6.6 Server Registers a New Treeconnect

Article02/14/2019

The CIFS or SMB2 server requesting registration of a **TreeConnect** MUST provide the tuple <**ServerName**, **ShareName**>. The server MUST insert a new **TreeConnect** into **TreeConnectList** and MUST assign **TreeConnect.GlobalTreeConnectId** the value that uniquely identifies the entry in the list. This value MUST be returned to the caller. The server MUST look up the **Share** in the **ShareList**, where **ShareName** matches **Share.ShareName**, and MUST increase **Share.CurrentUses** by 1.

## 3.1.6.7 Server Deregisters a Treeconnect

Article02/14/2019

The CIFS or SMB2 server MUST provide the tuple <ServerName, ShareName> and the *TreeconnectId* of the TreeConnect that is being deregistered.

The server MUST look up the TreeConnect in **TreeConnectList**, where **TreeConnect.GlobalTreeConnectId** is equal to the *TreeconnectId* provided by the caller, and MUST remove it from **TreeConnectList**. The server MUST look up the **Share** in the **ShareList**, where **ShareName** matches **Share.ShareName**, and MUST decrease **Share.CurrentUses** by 1.

## 3.1.6.8 Server Normalizes a ServerName

Article04/06/2021

The server MUST provide the tuple *<ServerName, ShareName>* as input parameters.

**ShareName:** The name of a shared resource.

**ServerName:** The name of a local server that the client is connecting to. This name MUST be less than 256 characters in length, and it MUST be a NetBIOS name, a fully qualified domain name (FQDN), a textual IPv4 or IPv6 address, or an empty string.

If ServerName is a nonempty string and it does not match any Transport.ServerName in **TransportList** and Alias.alias in **AliasList**, the server MUST set it as DefaultServerName. If ServerName is an empty string, the server MUST set it as "\*" to indicate that the local server name used.

If ShareName is empty, the server MUST determine the normalized ServerName to be returned using the following algorithm:

```
FOREACH Transport in TransportList
    IF ServerName is equal to Transport.ServerName
        RETURN ServerName
    ENDIF
ENDFOR
FOREACH Alias in AliasList
    IF ServerName is equal to Alias.alias
        RETURN Alias.target
    ENDIF
ENDFOR
RETURN DefaultServerName
```

If ShareName is not empty, to determine the normalized ServerName to be returned, the server MUST look up the share in **ShareList**, using the following algorithm:

```
FOREACH Share in ShareList
    IF ShareName is equal to Share.Name
        IF Share.ServerName is equal to ServerName
            RETURN Share.ServerName
        ELSE
            FOREACH Alias in AliasList
                IF ServerName is equal to Alias.alias
                    RETURN Alias.target
                ENDIF
            ENDFOR
        ENDIF
    ENDIF
```

```
ENDIF  
ENDFOR  
RETURN empty string
```

## 3.1.6.9 Local Application Enables Advertising a Service

Article02/14/2019

The caller MUST provide the service type flags, as specified in section [2.2.2.7](#), that it is enabling. The server MUST set these flag to TRUE in **GlobalServerAnnounce**.

## 3.1.6.10 Local Application Disables Advertising a Service

Article02/14/2019

The caller MUST provide the service type flags, as specified in section [2.2.2.7](#), that it is disabling. The server MUST set these flag to FALSE in **GlobalServerAnnounce**.

## 3.1.6.11 Server Queries Existing Services

Article02/14/2019

The server MUST return **GlobalServerAnnounce** to the caller to indicate the available services running on the server.

## 3.1.6.12 Server Service Terminates

Article02/14/2019

When the server service terminates, the server MUST disable the SMB server as specified in [\[MS-CIFS\]](#) section [3.3.4.19](#), and MUST disable the SMB2 server as specified in [\[MS-SMB2\]](#) section [3.3.4.23](#).

The server MUST remove all elements from **AliasList**, **ShareList**, and **TransportList**.

The server MUST free **AliasList**, **FileList**, **ShareList**, **SessionList**, **TransportList**, and **TreeConnectList**.

## 3.1.6.13 Local Application Pauses or Resumes the CIFS Server

Article06/24/2021

The server SHOULD <179> enforce security measures to verify that the caller has the required permissions to execute this routine. If the caller does not have the required credentials, the server SHOULD <180> fail the call. If the call is for the service to be paused, the server service MUST pause the CIFS server as specified in [MS-CIFS] section 3.3.4.20. If the call is for the service to be resumed, the server service MUST resume normal operation of the CIFS server as specified in [MS-CIFS] section 3.3.4.21.

## 3.1.6.14 Server Notifies Completion of Initialization

Article02/14/2019

The CIFS, SMB, or SMB2 server that calls this event provides a string that indicates the name of the protocol. If the protocol name is "CIFS", indicating notification from a CIFS or SMB server, the server MUST set **CifsInitialized** to TRUE. If the protocol name is "SMB2", the server MUST set **Smb2Initialized** to TRUE.

## 3.1.6.15 Server Notifies Current Uses of a Share

Article02/14/2019

The CIFS or SMB2 server MUST provide the tuple <ServerName, ShareName>. The server MUST look up the **Share** in the **ShareList**, where **ShareName** matches **Share.ShareName**, and MUST return **Share.CurrentUses**.

## 3.1.6.16 Server Updates Connection Count on a Transport

Article02/14/2019

The CIFS or SMB2 server MUST provide the tuple <TransportName,ConnectionFlag>. The server MUST look up the **Transport** in the **TransportList**, where TransportName matches **Transport.Name**. If ConnectionFlag is TRUE, the server MUST increase **Transport.ConnectionCount** by 1. If ConnectionFlag is FALSE, the server MUST decrease **Transport.ConnectionCount** by 1.

## 3.1.6.17 Server Looks Up Null Session Pipes

Article02/14/2019

The CIFS or SMB2 server MUST provide the pipe name, without the "\pipe\" prefix. The server MUST look up the pipe name in **NullSessionPipeList**. If a matching name is found in **NullSessionPipeList**, the server MUST return TRUE; otherwise, it MUST return FALSE.

## 3.2 Client Details

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [3.2.1 Abstract Data Model](#)
- [3.2.2 Timers](#)
- [3.2.3 Initialization](#)
- [3.2.4 Message Processing Events and Sequencing Rules](#)
- [3.2.5 Timer Events](#)
- [3.2.6 Other Local Events](#)

## 3.2.1 Abstract Data Model

Article02/14/2019

No abstract data model is used.

## 3.2.2 Timers

Article02/14/2019

No protocol timers are required beyond those internal ones that are used in [RPC](#) to implement resiliency to network outages. For more information, see [\[MS-RPCE\]](#).

### 3.2.3 Initialization

Article02/14/2019

The [client](#) MUST create an [RPC](#) connection to the remote computer, as specified in section [2.1](#).

## 3.2.4 Message Processing Events and Sequencing Rules

Article 06/24/2021

Upon the completion of the [RPC](#) method, the [client](#) MUST return the result unmodified to the higher layer. This is a stateless protocol with the exception of the [NetrShareDelCommit](#) method.

No sequence of method calls is imposed on this protocol, with the following exceptions:

1. NetrShareDelCommit method: The first phase MUST be completed (by the [NetrShareDelStart](#) method) before the second phase is attempted.
2. [NetrFileGetInfo](#) method: The [NetrFileEnum](#) method MUST be called to obtain the *FileId* before the [NetrFileGetInfo](#) method is called.
3. [NetrFileClose](#) method: [NetrFileEnum](#) MUST be called to obtain the *FileId* before the [NetrFileClose](#) method is called.

When a method is completed, the values that the RPC returns MUST be returned unmodified to the upper layer.

The client MUST ignore errors returned from the RPC [server](#) and notify the application invoker about the error that was received in the higher layer. Otherwise, no special message processing is required on the client beyond the processing that is required in the underlying RPC protocol.

## 3.2.5 Timer Events

Article02/14/2019

None.

## 3.2.6 Other Local Events

Article02/14/2019

None.

# 4 Protocol Examples

Article 06/24/2021

For most methods, the Server Service Remote Protocol is a simple request-response protocol. For every method that the [server](#) receives, except the [NetrShareDelStart](#) method and the [NetrShareDelCommit](#) method, the server executes the method and returns a completion. The [client](#) simply returns the completion status to the caller.

For example, the client calls the [NetrShareAdd](#) method, and the server executes the method and returns NERR\_Success, as shown in the following figure.

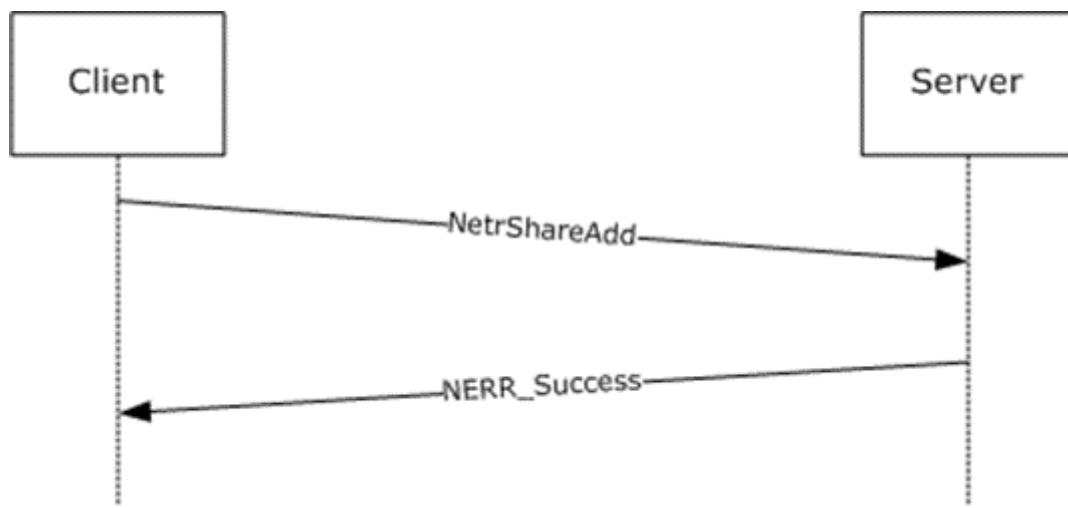


Figure 1: A simple request-response example

# 4.1 Example of ResumeHandle

Article 02/14/2019

The [client](#) calls the [NetrFileEnum](#) method to enumerate all open files on a [server](#) named "wingtiptoys". There are five open files on the server "wingtiptoys".

The client calls the NetrFileEnum method with the *ServerName* parameter equal to "wingtiptoys", and the *Level* field of the [FILE\\_ENUM\\_STRUCT](#) structure that is passed in the *InfoStruct* parameter is set to 0x00000003. The client also sets the *PreferredMaximumLength* parameter to 0x00000100 and passes a non-NULL pointer in the *TotalEntries* parameter and the *ResumeHandle* parameter.

If, for example, only the information for the first two open files fits into 0x00000100 bytes, when the server receives this method, it executes the method locally and returns [ERROR\\_MORE\\_DATA](#). The server returns the information for the first two open files in the *InfoStruct* parameter. It also sets the value of *TotalEntries* to 0x00000005 and the value of *ResumeHandle* to 0x00000120. The value of *ResumeHandle* is implementation-specific.

To continue enumerating the open files, the client calls the NetrFileEnum method with *ServerName* equal to "wingtiptoys", and the *Level* field of the [FILE\\_ENUM\\_STRUCT](#) structure that is passed in the *InfoStruct* parameter is set to 0x00000003. The client also sets the *PreferredMaximumLength* parameter to [MAX\\_PREFERRED\\_LENGTH](#) and passes a non-NULL pointer as *TotalEntries*. The client also passes the unchanged value of *ResumeHandle* (0x000000120).

On receiving this method, the server executes the method locally to continue enumeration based on a *ResumeHandle* value of 0x00000120 and returns [ERROR\\_SUCCESS](#). The server returns the names of the next three open files in the *InfoStruct* parameter. It also sets the value of *TotalEntries* to 0x00000003. The value of *ResumeHandle* is irrelevant.

## 4.2 Two-Phase Share Deletion

Article02/14/2019

The following figure shows the protocol message sequence for a two-phase share deletion.

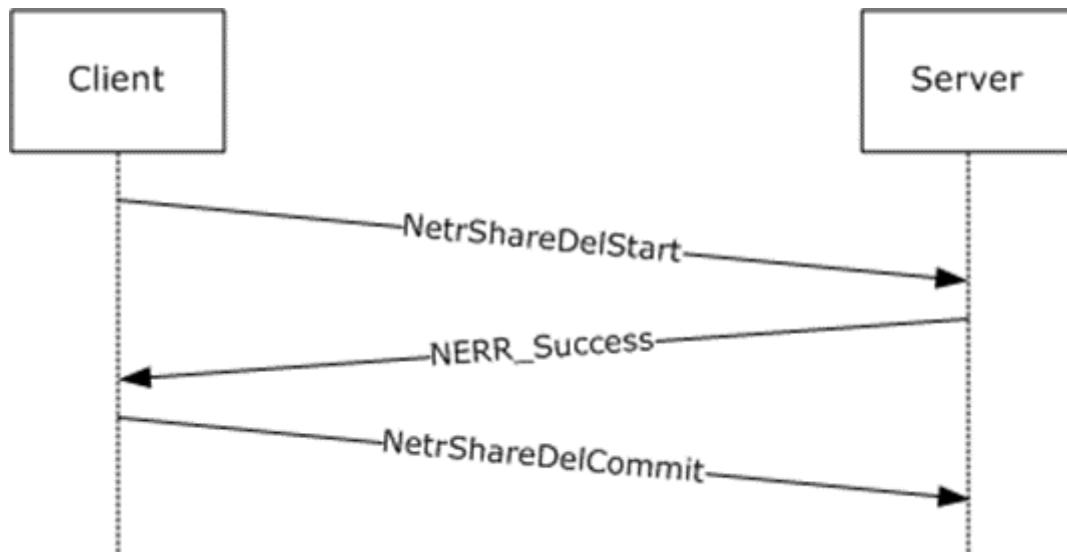


Figure 2: Two-phase share deletion

If the IPC\$ share is being deleted, a two-phase delete has to be performed because this action deletes the means of communication between the [client](#) and the [server](#). The following is the sequence of messages for a two-phase share delete:

1. The client sends the [NetrShareDelStart](#) method to the server.
2. The server processes the first phase of the delete and returns the status [NERR\\_Success](#).
3. The client sends the [NetrShareDelCommit](#) method to the server.
4. The server processes the second phase of the delete. Because the communication channel between the client and the server is deleted, the client does not receive a status that indicates the successful completion of the [NetrShareDelCommit](#) method.

# 4.3 Adding a Scoped Share With an Alias to a Server

Article 06/24/2021

The following figure shows the protocol message sequence for an administrator remotely configuring a server to support an additional server name, and configuring an alias for that new name.

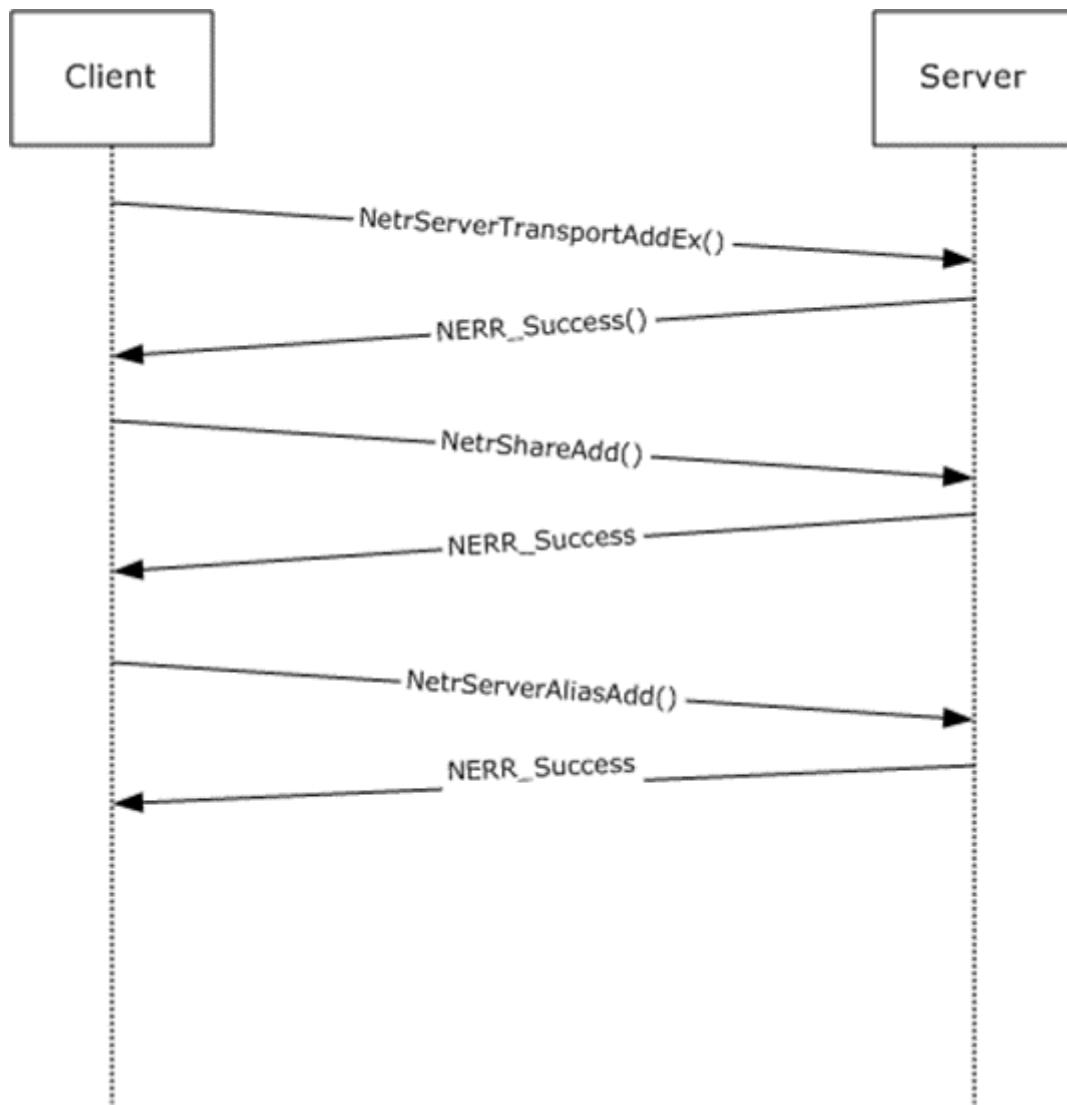


Figure 3: Message sequence for adding a scoped share with an alias to a server

1. The client calls `NetrServerTransportAddEx` (Opnum 41) to bind the server to the transport protocol with `svti3_transport_address` set to "server", and `SVTI2_SCOPED_NAME` set to TRUE.
2. The server processes the transport add and returns the status `NERR_Success`.
3. The client calls `NetrShareAdd` (Opnum 14) to add a share on the server. Along with other share parameters, the `shi303_servername` field is set to "server".

4. The server processes the share add and returns the status NERR\_Success.
5. The client calls NetrServerAliasAdd (Opnum 54) to add an alias, with srvai0\_alias set to "server.example.com", srvai0\_target set to "server", and srvai0\_default set to FALSE.
6. The server processes the alias add, and returns the status NERR\_Success.

On completion of these steps, a client connecting to the server and attempting to enumerate shares on this server and passing in "server" or "server.example.com" for the *ServerName* parameter for [NetrShareEnum](#), would find only those shares that were added as specified in step 3 above. Clients connecting and attempting to enumerate shares on this server and passing in any other name for the *ServerName* parameter for NetrShareEnum would not see the shares added as specified in step 3 above. (Note that the administrator is responsible for configuring the network such that the names "server" and "server.example.com" correctly resolve to the server above. This is not handled by NetrServerTransportAddEx (Opnum 41).)

# 5 Security

Article02/14/2019

This topic contains a number of related sections. The list below provides links to each section.

- [5.1 Security Considerations for Implementers](#)
- [5.2 Index of Security Parameters](#)

# 5.1 Security Considerations for Implementers

Article 02/14/2019

This protocol allows any user to connect to the [server](#); therefore, any security weakness in the server implementation could be exploitable. It is important that the server implementation enforce security on each method.

## 5.2 Index of Security Parameters

Article 02/14/2019

This protocol allows any user to establish a connection to the [RPC server](#) as specified in section [2.1](#).

# 6 Appendix A: Full IDL

Article10/04/2021

For ease of implementation, the full [IDL](#) is provided, where "ms-dtyp.idl" is the IDL as specified in [\[MS-DTYP\] Appendix A](#).

```
import "ms-dtyp.idl";

[
    uuid(4B324FC8-1670-01D3-1278-5A47BF6EE188),
    version(3.0),
    ms_union,
    pointer_default(unique)
]
interface srvsvc
{
    typedef [handle, string] wchar_t * SRVSVC_HANDLE;

    typedef struct _CONNECTION_INFO_0
    {
        DWORD coni0_id;
    } CONNECTION_INFO_0,
        *PCONNECTION_INFO_0,
        *LPCONNECTION_INFO_0;

    typedef struct _CONNECT_INFO_0_CONTAINER
    {
        DWORD EntriesRead;
        [size_is(EntriesRead)] LPCONNECTION_INFO_0 Buffer;
    } CONNECT_INFO_0_CONTAINER,
        *PCONNECT_INFO_0_CONTAINER,
        *LPCONNECT_INFO_0_CONTAINER;

    typedef struct _CONNECTION_INFO_1
    {
        DWORD coni1_id;
        DWORD coni1_type;
        DWORD coni1_numOpens;
        DWORD coni1_numUsers;
        DWORD coni1_time;
        [string] wchar_t * coni1_username;
        [string] wchar_t * coni1_netname;
    } CONNECTION_INFO_1,
        *PCONNECTION_INFO_1,
        *LPCONNECTION_INFO_1;

    typedef struct _CONNECT_INFO_1_CONTAINER
    {
        DWORD EntriesRead;
        [size_is(EntriesRead)] LPCONNECTION_INFO_1 Buffer;
    } CONNECT_INFO_1_CONTAINER,
```

```

*PCONNECT_INFO_1_CONTAINER,
*LPCONNECT_INFO_1_CONTAINER;

typedef [switch_type(DWORD)] union _CONNECT_ENUM_UNION {
    [case(0)]
        CONNECT_INFO_0_CONTAINER* Level0;
    [case(1)]
        CONNECT_INFO_1_CONTAINER* Level1;
} CONNECT_ENUM_UNION;

typedef struct _CONNECT_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] CONNECT_ENUM_UNION ConnectInfo;
} CONNECT_ENUM_STRUCT,
    *PCONNECT_ENUM_STRUCT,
    *LPCONNECT_ENUM_STRUCT;

typedef struct _FILE_INFO_2
{
    DWORD fi2_id;
} FILE_INFO_2, *PFILE_INFO_2, *LPFILE_INFO_2;

typedef struct _FILE_INFO_2_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPFILE_INFO_2 Buffer;
} FILE_INFO_2_CONTAINER,
    *PFILE_INFO_2_CONTAINER,
    *LPFILE_INFO_2_CONTAINER;

typedef struct _FILE_INFO_3 {
    DWORD fi3_id;
    DWORD fi3_permissions;
    DWORD fi3_num_locks;
    [string] wchar_t * fi3_pathname;
    [string] wchar_t * fi3_username;
} FILE_INFO_3,
    *PFILE_INFO_3,
    *LPFILE_INFO_3;

typedef struct _FILE_INFO_3_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPFILE_INFO_3 Buffer;
} FILE_INFO_3_CONTAINER,
    *PFILE_INFO_3_CONTAINER,
    *LPFILE_INFO_3_CONTAINER;

typedef [switch_type(DWORD)] union _FILE_ENUM_UNION {
    [case(2)]
        FILE_INFO_2_CONTAINER* Level2;
    [case(3)]
        FILE_INFO_3_CONTAINER* Level3;
} FILE_ENUM_UNION;

```

```

typedef struct _FILE_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] FILE_ENUM_UNION FileInfo;
} FILE_ENUM_STRUCT,
*PFILE_ENUM_STRUCT,
*LPFILE_ENUM_STRUCT;

typedef [switch_type(unsigned long)] union _FILE_INFO
{
    [case(2)]
        LPFILE_INFO_2 FileInfo2;
    [case(3)]
        LPFILE_INFO_3 FileInfo3;
} FILE_INFO,
*PFILE_INFO,
*LPFILE_INFO;

typedef struct _SESSION_INFO_0
{
    [string] wchar_t * sesi0_cname;
} SESSION_INFO_0,
*PSESSION_INFO_0,
*LPSESSION_INFO_0;

typedef struct _SESSION_INFO_0_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_0 Buffer;
} SESSION_INFO_0_CONTAINER,
*PSESSION_INFO_0_CONTAINER,
*LPSESSION_INFO_0_CONTAINER;

typedef struct _SESSION_INFO_1
{
    [string] wchar_t * sesi1_cname;
    [string] wchar_t * sesi1_username;
    DWORD sesi1_numOpens;
    DWORD sesi1_time;
    DWORD sesi1_idle_time;
    DWORD sesi1_user_flags;
} SESSION_INFO_1,
*PSESSION_INFO_1,
*LPSESSION_INFO_1;

typedef struct _SESSION_INFO_1_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_1 Buffer;
} SESSION_INFO_1_CONTAINER,
*PSESSION_INFO_1_CONTAINER,
*LPSESSION_INFO_1_CONTAINER;

typedef struct _SESSION_INFO_2
{
    [string] wchar_t * sesi2_cname;
}

```

```

        [string] wchar_t * sesi2_username;
        DWORD sesi2_numOpens;
        DWORD sesi2_time;
        DWORD sesi2_idle_time;
        DWORD sesi2_user_flags;
        [string] wchar_t * sesi2_cltype_name;
    } SESSION_INFO_2,
    *PSESSION_INFO_2,
    *LPSESSION_INFO_2;

typedef struct _SESSION_INFO_2_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_2 Buffer;
} SESSION_INFO_2_CONTAINER,
*PSESSION_INFO_2_CONTAINER,
*LPSESSION_INFO_2_CONTAINER;

typedef struct _SESSION_INFO_10
{
    [string] wchar_t * sesi10_cname;
    [string] wchar_t * sesi10_username;
    DWORD sesi10_time;
    DWORD sesi10_idle_time;
} SESSION_INFO_10,
*PSESSION_INFO_10,
*LPSESSION_INFO_10;

typedef struct _SESSION_INFO_10_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_10 Buffer;
} SESSION_INFO_10_CONTAINER,
*PSESSION_INFO_10_CONTAINER,
*LPSESSION_INFO_10_CONTAINER;

typedef struct _SESSION_INFO_502
{
    [string] wchar_t * sesi502_cname;
    [string] wchar_t * sesi502_username;
    DWORD sesi502_numOpens;
    DWORD sesi502_time;
    DWORD sesi502_idle_time;
    DWORD sesi502_user_flags;
    [string] wchar_t * sesi502_cltype_name;
    [string] wchar_t * sesi502_transport;
} SESSION_INFO_502,
*PSESSION_INFO_502,
*LPSESSION_INFO_502;

typedef struct _SESSION_INFO_502_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSESSION_INFO_502 Buffer;
} SESSION_INFO_502_CONTAINER,

```

```

*PSESSION_INFO_502_CONTAINER,
*LPSESSION_INFO_502_CONTAINER;

typedef [switch_type(DWORD)] union _SESSION_ENUM_UNION {
[case(0)]
    SESSION_INFO_0_CONTAINER* Level0;
[case(1)]
    SESSION_INFO_1_CONTAINER* Level1;
[case(2)]
    SESSION_INFO_2_CONTAINER* Level2;
[case(10)]
    SESSION_INFO_10_CONTAINER* Level10;
[case(502)]
    SESSION_INFO_502_CONTAINER* Level502;
} SESSION_ENUM_UNION;

typedef struct _SESSION_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] SESSION_ENUM_UNION SessionInfo;
} SESSION_ENUM_STRUCT,
*PSESSION_ENUM_STRUCT,
*LPSESSION_ENUM_STRUCT;

typedef struct _SHARE_INFO_502_I
{
    [string] WCHAR * shi502_netname;
    DWORD shi502_type;
    [string] WCHAR * shi502_remark;
    DWORD shi502_permissions;
    DWORD shi502_max_uses;
    DWORD shi502_current_uses;
    [string] WCHAR * shi502_path;
    [string] WCHAR * shi502_passwd;
    DWORD shi502_reserved;
    [size_is(shi502_reserved)] unsigned char
        * shi502_security_descriptor;
} SHARE_INFO_502_I,
*PSHARE_INFO_502_I,
*LPSHARE_INFO_502_I;

typedef struct _SHARE_INFO_503_I
{
    [string] WCHAR * shi503_netname;
    DWORD shi503_type;
    [string] WCHAR * shi503_remark;
    DWORD shi503_permissions;
    DWORD shi503_max_uses;
    DWORD shi503_current_uses;
    [string] WCHAR * shi503_path;
    [string] WCHAR * shi503_passwd;
    [string] WCHAR * shi503_servername;
    DWORD shi503_reserved;
    [size_is(shi503_reserved)] PUCHAR shi503_security_descriptor;
} SHARE_INFO_503_I,
*PSHARE_INFO_503_I,

```

```
*LPSHARE_INFO_503_I;

typedef struct _SHARE_INFO_503_CONTAINER
{   DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_503_I Buffer;
} SHARE_INFO_503_CONTAINER,
    *PSHARE_INFO_503_CONTAINER,
    *LPSHARE_INFO_503_CONTAINER;

typedef struct _SHARE_INFO_1501_I
{
    DWORD shi1501_reserved;
    [size_is(shi1501_reserved)] unsigned char
        * shi1501_security_descriptor;
} SHARE_INFO_1501_I,
    *PSHARE_INFO_1501_I,
    *LPSHARE_INFO_1501_I;

typedef struct _SHARE_INFO_0
{
    [string] wchar_t * shi0_netname;
} SHARE_INFO_0,
    *PSHARE_INFO_0,
    *LPSHARE_INFO_0;

typedef struct _SHARE_INFO_0_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_0 Buffer;
} SHARE_INFO_0_CONTAINER;

typedef struct _SHARE_INFO_1
{
    [string] wchar_t * shi1_netname;
    DWORD shi1_type;
    [string] wchar_t * shi1_remark;
} SHARE_INFO_1,
    *PSHARE_INFO_1,
    *LPSHARE_INFO_1;

typedef struct _SHARE_INFO_1_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_1 Buffer;
} SHARE_INFO_1_CONTAINER;

typedef struct _SHARE_INFO_2
{
    [string] wchar_t * shi2_netname;
    DWORD shi2_type;
    [string] wchar_t * shi2_remark;
    DWORD shi2_permissions;
    DWORD shi2_max_uses;
    DWORD shi2_current_uses;
    [string] wchar_t * shi2_path;
```

```

        [string] wchar_t * shi2_passwd;
    } SHARE_INFO_2,
    *PSHARE_INFO_2,
    *LPSHARE_INFO_2;

typedef struct _SHARE_INFO_2_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_2 Buffer;
} SHARE_INFO_2_CONTAINER,
*PSHARE_INFO_2_CONTAINER,
*LPSHARE_INFO_2_CONTAINER;

typedef struct _SHARE_INFO_501
{
    [string] wchar_t * shi501_netname;
    DWORD shi501_type;
    [string] wchar_t * shi501_remark;
    DWORD shi501_flags;
} SHARE_INFO_501,
*PSHARE_INFO_501,
*LPSHARE_INFO_501;

typedef struct _SHARE_INFO_501_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_501 Buffer;
} SHARE_INFO_501_CONTAINER, *PSHARE_INFO_501_CONTAINER,
*LPSHARE_INFO_501_CONTAINER;

typedef struct _SHARE_INFO_502_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSHARE_INFO_502_I Buffer;
} SHARE_INFO_502_CONTAINER,
*PSHARE_INFO_502_CONTAINER,
*LPSHARE_INFO_502_CONTAINER;

typedef [switch_type(DWORD)] union _SHARE_ENUM_UNION {
    [case(0)]
        SHARE_INFO_0_CONTAINER* Level0;
    [case(1)]
        SHARE_INFO_1_CONTAINER* Level1;
    [case(2)]
        SHARE_INFO_2_CONTAINER* Level2;
    [case(501)]
        SHARE_INFO_501_CONTAINER* Level501;
    [case(502)]
        SHARE_INFO_502_CONTAINER* Level502;
    [case(503)]
        SHARE_INFO_503_CONTAINER* Level503;
} SHARE_ENUM_UNION;

typedef struct _SHARE_ENUM_STRUCT
{

```

```

        DWORD Level;
        [switch_is(Level)] SHARE_ENUM_UNION ShareInfo;
    } SHARE_ENUM_STRUCT,
        *PSHARE_ENUM_STRUCT,
        *LPSHARE_ENUM_STRUCT;

typedef struct _SHARE_INFO_1004
{
    [string] wchar_t * shi1004_remark;
} SHARE_INFO_1004,
    *PSHARE_INFO_1004,
    *LPSHARE_INFO_1004;

typedef struct _SHARE_INFO_1006
{
    DWORD shi1006_max_uses;
} SHARE_INFO_1006,
    *PSHARE_INFO_1006,
    *LPSHARE_INFO_1006;

typedef struct _SHARE_INFO_1005
{
    DWORD shi1005_flags;
} SHARE_INFO_1005,
    *PSHARE_INFO_1005,
    *LPSHARE_INFO_1005;

//JMP: order differs in documentation
typedef [switch_type(unsigned long)] union _SHARE_INFO
// for Get & Set info
{
    [case(0)]
        LPSHARE_INFO_0 ShareInfo0;
    [case(1)]
        LPSHARE_INFO_1 ShareInfo1;
    [case(2)]
        LPSHARE_INFO_2 ShareInfo2;
    [case(502)]
        LPSHARE_INFO_502_I ShareInfo502;
    [case(1004)]
        LPSHARE_INFO_1004 ShareInfo1004;
    [case(1006)]
        LPSHARE_INFO_1006 ShareInfo1006;
    [case(1501)]
        LPSHARE_INFO_1501_I ShareInfo1501;
    [default]
        ;
    [case(1005)]
        LPSHARE_INFO_1005 ShareInfo1005;
    [case(501)]
        LPSHARE_INFO_501 ShareInfo501;
    [case(503)]
        LPSHARE_INFO_503_I ShareInfo503;
} SHARE_INFO,
    *PSHARE_INFO,

```

```
*LPSHARE_INFO;

typedef struct _SERVER_INFO_100
{
    DWORD sv100_platform_id;
    [string] wchar_t* sv100_name;
} SERVER_INFO_100,
*PSERVER_INFO_100,
*LPSERVER_INFO_100;

typedef struct _SERVER_INFO_101
{
    DWORD sv101_platform_id;
    [string] wchar_t* sv101_name;
    DWORD sv101_version_major;
    DWORD sv101_version_minor;
    DWORD sv101_type;
    [string] wchar_t* sv101_comment;
} SERVER_INFO_101,
*PSERVER_INFO_101,
*LPSERVER_INFO_101;

typedef struct _SERVER_INFO_102
{
    DWORD sv102_platform_id;
    [string] wchar_t * sv102_name;
    DWORD sv102_version_major;
    DWORD sv102_version_minor;
    DWORD sv102_type;
    [string] wchar_t * sv102_comment;
    DWORD sv102_users;
    long sv102_disc;
    int sv102_hidden;
    DWORD sv102_announce;
    DWORD sv102_anndelta;
    DWORD sv102_licenses;
    [string] wchar_t * sv102_userpath;
} SERVER_INFO_102,
*PSERVER_INFO_102,
*LPSERVER_INFO_102;

typedef struct _SERVER_INFO_103
{
    DWORD sv103_platform_id;
    [string] wchar_t* sv103_name;
    DWORD sv103_version_major;
    DWORD sv103_version_minor;
    DWORD sv103_type;
    [string] wchar_t* sv103_comment;
    DWORD sv103_users;
    LONG sv103_disc;
    BOOL sv103_hidden;
    DWORD sv103_announce;
```

```
        DWORD sv103_anndelta;
        DWORD sv103_licenses;
        [string] wchar_t* sv103_userpath;
        DWORD sv103_capabilities;
    } SERVER_INFO_103,
*PSERVER_INFO_103,
*LPSERVER_INFO_103;

typedef struct _SERVER_INFO_502
{
    DWORD sv502_sessopens;
    DWORD sv502_sessvcs;
    DWORD sv502_opensearch;
    DWORD sv502_sizreqbuf;
    DWORD sv502_initworkitems;
    DWORD sv502_maxworkitems;
    DWORD sv502_rawworkitems;
    DWORD sv502_irpstacksize;
    DWORD sv502_maxrawbuflen;
    DWORD sv502_sessusers;
    DWORD sv502_sessconns;
    DWORD sv502_maxpagedmemoryusage;
    DWORD sv502_maxnonpagedmemoryusage;
    int sv502_enablessoftcompat;
    int sv502_enableforcedlogoff;
    int sv502_timesource;
    int sv502_acceptdownlevelapis;
    int sv502_lmannounce;
} SERVER_INFO_502,
*PSERVER_INFO_502,
*LPSERVER_INFO_502;

typedef struct _SERVER_INFO_503
{
    DWORD sv503_sessopens;
    DWORD sv503_sessvcs;
    DWORD sv503_opensearch;
    DWORD sv503_sizreqbuf;
    DWORD sv503_initworkitems;
    DWORD sv503_maxworkitems;
    DWORD sv503_rawworkitems;
    DWORD sv503_irpstacksize;
    DWORD sv503_maxrawbuflen;
    DWORD sv503_sessusers;
    DWORD sv503_sessconns;
    DWORD sv503_maxpagedmemoryusage;
    DWORD sv503_maxnonpagedmemoryusage;
    int sv503_enablessoftcompat;
    int sv503_enableforcedlogoff;
    int sv503_timesource;
    int sv503_acceptdownlevelapis;
    int sv503_lmannounce;
    [string] wchar_t * sv503_domain;
    DWORD sv503_maxcopyreadlen;
    DWORD sv503_maxcopywritelen;
```

```
    DWORD sv503_minkeepsearch;
    DWORD sv503_maxkeepsearch;
    DWORD sv503_minkeepcomplsearch;
    DWORD sv503_maxkeepcomplsearch;
    DWORD sv503_threadcountadd;
    DWORD sv503_numblockthreads;
    DWORD sv503_scavtimeout;
    DWORD sv503_minrcvqueue;
    DWORD sv503_minfreeworkitems;
    DWORD sv503_xactmemsize;
    DWORD sv503_threadpriority;
    DWORD sv503_maxmpxct;
    DWORD sv503_oplockbreakwait;
    DWORD sv503_oplockbreakresponsewait;
    int sv503_enableoplocks;
    int sv503_enableoplockforceclose;
    int sv503_enablefcopens;
    int sv503_enableraw;
    int sv503_enablessharednetdrives;
    DWORD sv503_minfreeconnections;
    DWORD sv503_maxfreeconnections;
} SERVER_INFO_503,
*PSERVER_INFO_503,
*LPSERVER_INFO_503;
```

```
typedef struct _SERVER_INFO_599
{
    DWORD sv599_sessopens;
    DWORD sv599_sessvcs;
    DWORD sv599_opensearch;
    DWORD sv599_sizreqbuf;
    DWORD sv599_initworkitems;
    DWORD sv599_maxworkitems;
    DWORD sv599_rawworkitems;
    DWORD sv599_irpstacksize;
    DWORD sv599_maxrawbuflen;
    DWORD sv599_sessusers;
    DWORD sv599_sessconns;
    DWORD sv599_maxpagedmemoryusage;
    DWORD sv599_maxnonpagedmemoryusage;
    int sv599_enablessoftcompat;
    int sv599_enableforcedlogoff;
    int sv599_timeresource;
    int sv599_acceptdownlevelapis;
    int sv599_lmannounce;
    [string] wchar_t * sv599_domain;
    DWORD sv599_maxcopyreadlen;
    DWORD sv599_maxcopywritelen;
    DWORD sv599_minkeepsearch;
    DWORD sv599_maxkeepsearch;
    DWORD sv599_minkeepcomplsearch;
    DWORD sv599_maxkeepcomplsearch;
    DWORD sv599_threadcountadd;
    DWORD sv599_numblockthreads;
    DWORD sv599_scavtimeout;
```

```

        DWORD sv599_minrcvqueue;
        DWORD sv599_minfreeworkitems;
        DWORD sv599_xactmemsize;
        DWORD sv599_threadpriority;
        DWORD sv599_maxmpxct;
        DWORD sv599_oplockbreakwait;
        DWORD sv599_oplockbreakresponsewait;
        int sv599_enableoplocks;
        int sv599_enableoplockforceclose;
        int sv599_enablefcopens;
        int sv599_enableraw;
        int sv599_enablessharednetdrives;
        DWORD sv599_minfreeconnections;
        DWORD sv599_maxfreeconnections;
        DWORD sv599_initsesstable;
        DWORD sv599_initconntable;
        DWORD sv599_initfiletable;
        DWORD sv599_initsearchtable;
        DWORD sv599_alertschedule;
        DWORD sv599_errorthreshold;
        DWORD sv599_networkerrorthreshold;
        DWORD sv599_diskspacethreshold;
        DWORD sv599_reserved;
        DWORD sv599_maxlinkdelay;
        DWORD sv599_minlinkthroughput;
        DWORD sv599_linkinfovalidtime;
        DWORD sv599_scavqosinfoupdatedatetime;
        DWORD sv599_maxworkitemidletime;
    } SERVER_INFO_599,
    *PSERVER_INFO_599,
    *LPSERVER_INFO_599;

typedef struct _SERVER_INFO_1005
{
    [string] wchar_t * sv1005_comment;
} SERVER_INFO_1005,
*PSERVER_INFO_1005,
*LPSERVER_INFO_1005;

typedef struct _SERVER_INFO_1107
{
    DWORD sv1107_users;
} SERVER_INFO_1107,
*PSERVER_INFO_1107,
*LPSERVER_INFO_1107;

typedef struct _SERVER_INFO_1010
{
    long sv1010_disc;
} SERVER_INFO_1010,
*PSERVER_INFO_1010,
*LPSERVER_INFO_1010;

typedef struct _SERVER_INFO_1016
{

```

```
    int sv1016_hidden;
} SERVER_INFO_1016,
 *PSERVER_INFO_1016,
 *LPSERVER_INFO_1016;

typedef struct _SERVER_INFO_1017
{
    DWORD sv1017_announce;
} SERVER_INFO_1017,
 *PSERVER_INFO_1017,
 *LPSERVER_INFO_1017;

typedef struct _SERVER_INFO_1018
{
    DWORD sv1018_anndelta;
} SERVER_INFO_1018,
 *PSERVER_INFO_1018,
 *LPSERVER_INFO_1018;

typedef struct _SERVER_INFO_1501
{
    DWORD sv1501_sessopens;
} SERVER_INFO_1501,
 *PSERVER_INFO_1501,
 *LPSERVER_INFO_1501;

typedef struct _SERVER_INFO_1502
{
    DWORD sv1502_sessvcs;
} SERVER_INFO_1502,
 *PSERVER_INFO_1502,
 *LPSERVER_INFO_1502;

typedef struct _SERVER_INFO_1503
{
    DWORD sv1503_opensearch;
} SERVER_INFO_1503, *PSERVER_INFO_1503, *LPSERVER_INFO_1503;

typedef struct _SERVER_INFO_1506
{
    DWORD sv1506_maxworkitems;
} SERVER_INFO_1506, *PSERVER_INFO_1506, *LPSERVER_INFO_1506;

typedef struct _SERVER_INFO_1510
{
    DWORD sv1510_sessusers;
} SERVER_INFO_1510, *PSERVER_INFO_1510, *LPSERVER_INFO_1510;

typedef struct _SERVER_INFO_1511
{
    DWORD sv1511_sessconns;
} SERVER_INFO_1511, *PSERVER_INFO_1511, *LPSERVER_INFO_1511;

typedef struct _SERVER_INFO_1512
{
```

```
        DWORD sv1512_maxnonpagedmemoryusage;
} SERVER_INFO_1512, *PSERVER_INFO_1512, *LPSERVER_INFO_1512;

typedef struct _SERVER_INFO_1513
{
    DWORD sv1513_maxpagedmemoryusage;
} SERVER_INFO_1513, *PSERVER_INFO_1513, *LPSERVER_INFO_1513;

typedef struct _SERVER_INFO_1514
{
    int sv1514_enablesoftcompat;
} SERVER_INFO_1514, *PSERVER_INFO_1514, *LPSERVER_INFO_1514;

typedef struct _SERVER_INFO_1515
{
    int sv1515_enableforcedlogoff;
} SERVER_INFO_1515, *PSERVER_INFO_1515, *LPSERVER_INFO_1515;

typedef struct _SERVER_INFO_1516
{
    int sv1516_timesource;
} SERVER_INFO_1516, *PSERVER_INFO_1516, *LPSERVER_INFO_1516;

typedef struct _SERVER_INFO_1518
{
    int sv1518_lmannounce;
} SERVER_INFO_1518, *PSERVER_INFO_1518, *LPSERVER_INFO_1518;

typedef struct _SERVER_INFO_1523
{
    DWORD sv1523_maxkeepsearch;
} SERVER_INFO_1523, *PSERVER_INFO_1523, *LPSERVER_INFO_1523;

typedef struct _SERVER_INFO_1528
{
    DWORD sv1528_scavtimeout;
} SERVER_INFO_1528, *PSERVER_INFO_1528, *LPSERVER_INFO_1528;

typedef struct _SERVER_INFO_1529
{
    DWORD sv1529_minrcvqueue;
} SERVER_INFO_1529, *PSERVER_INFO_1529, *LPSERVER_INFO_1529;

typedef struct _SERVER_INFO_1530
{
    DWORD sv1530_minfreeworkitems;
} SERVER_INFO_1530, *PSERVER_INFO_1530, *LPSERVER_INFO_1530;

typedef struct _SERVER_INFO_1533
{
    DWORD sv1533_maxmpxct;
} SERVER_INFO_1533, *PSERVER_INFO_1533, *LPSERVER_INFO_1533;

typedef struct _SERVER_INFO_1534
{
```

```
        DWORD sv1534_oplockbreakwait;
} SERVER_INFO_1534, *PSERVER_INFO_1534, *LPSERVER_INFO_1534;

typedef struct _SERVER_INFO_1535
{
    DWORD sv1535_oplockbreakresponsewait;
} SERVER_INFO_1535, *PSERVER_INFO_1535, *LPSERVER_INFO_1535;

typedef struct _SERVER_INFO_1536
{
    int sv1536_enableoplocks;
} SERVER_INFO_1536, *PSERVER_INFO_1536, *LPSERVER_INFO_1536;

typedef struct _SERVER_INFO_1538
{
    int sv1538_enablefcbopens;
} SERVER_INFO_1538, *PSERVER_INFO_1538, *LPSERVER_INFO_1538;

typedef struct _SERVER_INFO_1539
{
    int sv1539_enableraw;
} SERVER_INFO_1539, *PSERVER_INFO_1539, *LPSERVER_INFO_1539;

typedef struct _SERVER_INFO_1540
{
    int sv1540_enablessharednetdrives;
} SERVER_INFO_1540, *PSERVER_INFO_1540, *LPSERVER_INFO_1540;

typedef struct _SERVER_INFO_1541
{
    int sv1541_minfreeconnections;
} SERVER_INFO_1541, *PSERVER_INFO_1541, *LPSERVER_INFO_1541;

typedef struct _SERVER_INFO_1542
{
    int sv1542_maxfreeconnections;
} SERVER_INFO_1542, *PSERVER_INFO_1542, *LPSERVER_INFO_1542;

typedef struct _SERVER_INFO_1543
{
    DWORD sv1543_initsesstable;
} SERVER_INFO_1543, *PSERVER_INFO_1543, *LPSERVER_INFO_1543;

typedef struct _SERVER_INFO_1544
{
    DWORD sv1544_initconntable;
} SERVER_INFO_1544, *PSERVER_INFO_1544, *LPSERVER_INFO_1544;

typedef struct _SERVER_INFO_1545
{
    DWORD sv1545_initfiletable;
} SERVER_INFO_1545, *PSERVER_INFO_1545, *LPSERVER_INFO_1545;

typedef struct _SERVER_INFO_1546
{
```

```

        DWORD sv1546_initsearchtable;
} SERVER_INFO_1546, *PSERVER_INFO_1546, *LPSERVER_INFO_1546;

typedef struct _SERVER_INFO_1547
{
    DWORD sv1547_alertschedule;
} SERVER_INFO_1547, *PSERVER_INFO_1547, *LPSERVER_INFO_1547;

typedef struct _SERVER_INFO_1548
{
    DWORD sv1548_errorthreshold;
} SERVER_INFO_1548, *PSERVER_INFO_1548, *LPSERVER_INFO_1548;

typedef struct _SERVER_INFO_1549
{
    DWORD sv1549_networkerrorthreshold;
} SERVER_INFO_1549, *PSERVER_INFO_1549, *LPSERVER_INFO_1549;

typedef struct _SERVER_INFO_1550
{
    DWORD sv1550_diskspacethreshold;
} SERVER_INFO_1550, *PSERVER_INFO_1550, *LPSERVER_INFO_1550;

typedef struct _SERVER_INFO_1552
{
    DWORD sv1552_maxlinkdelay;
} SERVER_INFO_1552, *PSERVER_INFO_1552, *LPSERVER_INFO_1552;

typedef struct _SERVER_INFO_1553
{
    DWORD sv1553_minlinkthroughput;
} SERVER_INFO_1553, *PSERVER_INFO_1553, *LPSERVER_INFO_1553;

typedef struct _SERVER_INFO_1554
{
    DWORD sv1554_linkinfovalidtime;
} SERVER_INFO_1554, *PSERVER_INFO_1554, *LPSERVER_INFO_1554;

typedef struct _SERVER_INFO_1555
{
    DWORD sv1555_scavqosinfoupdatetime;
} SERVER_INFO_1555, *PSERVER_INFO_1555, *LPSERVER_INFO_1555;

typedef struct _SERVER_INFO_1556
{
    DWORD sv1556_maxworkitemidletime;
} SERVER_INFO_1556, *PSERVER_INFO_1556, *LPSERVER_INFO_1556;

typedef [switch_type(unsigned long)] union _SERVER_INFO
{
    [case(100)]
        LPSERVER_INFO_100 ServerInfo100;
    [case(101)]
        LPSERVER_INFO_101 ServerInfo101;
    [case(102)]

```

```
    LPSCALAR_INFO_102 ServerInfo102;
[case(103)]
    LPSCALAR_INFO_103 ServerInfo103;
[case(502)]
    LPSCALAR_INFO_502 ServerInfo502;
[case(503)]
    LPSCALAR_INFO_503 ServerInfo503;
[case(599)]
    LPSCALAR_INFO_599 ServerInfo599;
[case(1005)]
    LPSCALAR_INFO_1005 ServerInfo1005;
[case(1107)]
    LPSCALAR_INFO_1107 ServerInfo1107;
[case(1010)]
    LPSCALAR_INFO_1010 ServerInfo1010;
[case(1016)]
    LPSCALAR_INFO_1016 ServerInfo1016;
[case(1017)]
    LPSCALAR_INFO_1017 ServerInfo1017;
[case(1018)]
    LPSCALAR_INFO_1018 ServerInfo1018;
[case(1501)]
    LPSCALAR_INFO_1501 ServerInfo1501;
[case(1502)]
    LPSCALAR_INFO_1502 ServerInfo1502;
[case(1503)]
    LPSCALAR_INFO_1503 ServerInfo1503;
[case(1506)]
    LPSCALAR_INFO_1506 ServerInfo1506;
[case(1510)]
    LPSCALAR_INFO_1510 ServerInfo1510;
[case(1511)]
    LPSCALAR_INFO_1511 ServerInfo1511;
[case(1512)]
    LPSCALAR_INFO_1512 ServerInfo1512;
[case(1513)]
    LPSCALAR_INFO_1513 ServerInfo1513;
[case(1514)]
    LPSCALAR_INFO_1514 ServerInfo1514;
[case(1515)]
    LPSCALAR_INFO_1515 ServerInfo1515;
[case(1516)]
    LPSCALAR_INFO_1516 ServerInfo1516;
[case(1518)]
    LPSCALAR_INFO_1518 ServerInfo1518;
[case(1523)]
    LPSCALAR_INFO_1523 ServerInfo1523;
[case(1528)]
    LPSCALAR_INFO_1528 ServerInfo1528;
[case(1529)]
    LPSCALAR_INFO_1529 ServerInfo1529;
[case(1530)]
    LPSCALAR_INFO_1530 ServerInfo1530;
[case(1533)]
    LPSCALAR_INFO_1533 ServerInfo1533;
```

```

        [case(1534)]
            LPSERVER_INFO_1534 ServerInfo1534;
        [case(1535)]
            LPSERVER_INFO_1535 ServerInfo1535;
        [case(1536)]
            LPSERVER_INFO_1536 ServerInfo1536;
        [case(1538)]
            LPSERVER_INFO_1538 ServerInfo1538;
        [case(1539)]
            LPSERVER_INFO_1539 ServerInfo1539;
        [case(1540)]
            LPSERVER_INFO_1540 ServerInfo1540;
        [case(1541)]
            LPSERVER_INFO_1541 ServerInfo1541;
        [case(1542)]
            LPSERVER_INFO_1542 ServerInfo1542;
        [case(1543)]
            LPSERVER_INFO_1543 ServerInfo1543;
        [case(1544)]
            LPSERVER_INFO_1544 ServerInfo1544;
        [case(1545)]
            LPSERVER_INFO_1545 ServerInfo1545;
        [case(1546)]
            LPSERVER_INFO_1546 ServerInfo1546;
        [case(1547)]
            LPSERVER_INFO_1547 ServerInfo1547;
        [case(1548)]
            LPSERVER_INFO_1548 ServerInfo1548;
        [case(1549)]
            LPSERVER_INFO_1549 ServerInfo1549;
        [case(1550)]
            LPSERVER_INFO_1550 ServerInfo1550;
        [case(1552)]
            LPSERVER_INFO_1552 ServerInfo1552;
        [case(1553)]
            LPSERVER_INFO_1553 ServerInfo1553;
        [case(1554)]
            LPSERVER_INFO_1554 ServerInfo1554;
        [case(1555)]
            LPSERVER_INFO_1555 ServerInfo1555;
        [case(1556)]
            LPSERVER_INFO_1556 ServerInfo1556;
    } SERVER_INFO, *PSERVER_INFO, *LPSERVER_INFO;

typedef struct _DISK_INFO
{
    [string] WCHAR Disk[3];
} DISK_INFO, *PDISK_INFO, *LPDISK_INFO;

typedef struct _DISK_ENUM_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead), length_is(EntriesRead)] LPDISK_INFO
        Buffer;
} DISK_ENUM_CONTAINER;

```

```

typedef struct _SERVER_TRANSPORT_INFO_0
{
    DWORD svti0_numberofvcs;
    [string] wchar_t * svti0_transportname;
    [size_is(svti0_transportaddresslength)] unsigned char
        * svti0_transportaddress;
    DWORD svti0_transportaddresslength;
    [string] wchar_t * svti0_networkaddress;
} SERVER_TRANSPORT_INFO_0, *PSERVER_TRANSPORT_INFO_0,
    *LPSERVER_TRANSPORT_INFO_0;

typedef struct _SERVER_XPORT_INFO_0_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSERVER_TRANSPORT_INFO_0 Buffer;
} SERVER_XPORT_INFO_0_CONTAINER, *PSERVER_XPORT_INFO_0_CONTAINER;

typedef struct _SERVER_TRANSPORT_INFO_1
{
    DWORD svti1_numberofvcs;
    [string] wchar_t * svti1_transportname;
    [size_is(svti1_transportaddresslength)] unsigned char
        * svti1_transportaddress;
    DWORD svti1_transportaddresslength;
    [string] wchar_t * svti1_networkaddress;
    [string] wchar_t * svti1_domain;
} SERVER_TRANSPORT_INFO_1, *PSERVER_TRANSPORT_INFO_1,
    *LPSERVER_TRANSPORT_INFO_1;

typedef struct _SERVER_XPORT_INFO_1_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSERVER_TRANSPORT_INFO_1 Buffer;
} SERVER_XPORT_INFO_1_CONTAINER, *PSERVER_XPORT_INFO_1_CONTAINER;

typedef struct _SERVER_TRANSPORT_INFO_2
{
    DWORD svti2_numberofvcs;
    [string] wchar_t * svti2_transportname;
    [size_is(svti2_transportaddresslength)] unsigned char
        * svti2_transportaddress;
    DWORD svti2_transportaddresslength;
    [string] wchar_t * svti2_networkaddress;
    [string] wchar_t * svti2_domain;
    unsigned long svti2_flags;
} SERVER_TRANSPORT_INFO_2, *PSERVER_TRANSPORT_INFO_2,
    *LPSERVER_TRANSPORT_INFO_2;

typedef struct _SERVER_XPORT_INFO_2_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSERVER_TRANSPORT_INFO_2 Buffer;
} SERVER_XPORT_INFO_2_CONTAINER, *PSERVER_XPORT_INFO_2_CONTAINER;

```

```

typedef struct _SERVER_TRANSPORT_INFO_3
{
    DWORD svti3_numberofvcs;
    [string] wchar_t * svti3_transportname;
    [size_is(svti3_transportaddresslength)] unsigned char
        * svti3_transportaddress;
    DWORD svti3_transportaddresslength;
    [string] wchar_t * svti3_networkaddress;
    [string] wchar_t * svti3_domain;
    unsigned long svti3_flags;
    DWORD svti3_passwordlength;
    unsigned char svti3_password[ 256 ];
} SERVER_TRANSPORT_INFO_3, *PSERVER_TRANSPORT_INFO_3,
    *LPSERVER_TRANSPORT_INFO_3;

typedef struct _SERVER_XPORT_INFO_3_CONTAINER
{
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPSERVER_TRANSPORT_INFO_3 Buffer;
} SERVER_XPORT_INFO_3_CONTAINER, *PSERVER_XPORT_INFO_3_CONTAINER;

typedef [switch_type(unsigned long)] union _TRANSPORT_INFO
{
    [case(0)]
        SERVER_TRANSPORT_INFO_0 Transport0;
    [case(1)]
        SERVER_TRANSPORT_INFO_1 Transport1;
    [case(2)]
        SERVER_TRANSPORT_INFO_2 Transport2;
    [case(3)]
        SERVER_TRANSPORT_INFO_3 Transport3;
} TRANSPORT_INFO, *PTRANSPORT_INFO, *LPTRANSPORT_INFO;

typedef [switch_type(DWORD)] union _SERVER_XPORT_ENUM_UNION {
    [case(0)]
        PSERVER_XPORT_INFO_0_CONTAINER Level0;
    [case(1)]
        PSERVER_XPORT_INFO_1_CONTAINER Level1;
    [case(2)]
        PSERVER_XPORT_INFO_2_CONTAINER Level2;
    [case(3)]
        PSERVER_XPORT_INFO_3_CONTAINER Level3;
} SERVER_XPORT_ENUM_UNION;

typedef struct _SERVER_XPORT_ENUM_STRUCT
{
    DWORD Level;
    [switch_is(Level)] SERVER_XPORT_ENUM_UNION XportInfo;
} SERVER_XPORT_ENUM_STRUCT, *PSERVER_XPORT_ENUM_STRUCT,
    *LPSERVER_XPORT_ENUM_STRUCT;

typedef [context_handle] void *SHARE_DEL_HANDLE;
typedef SHARE_DEL_HANDLE *PSHARE_DEL_HANDLE;

```

```

typedef struct _ADT_SECURITY_DESCRIPTOR
{
    DWORD Length;
    [size_is(Length)] unsigned char * Buffer;
} ADT_SECURITY_DESCRIPTOR, *PADT_SECURITY_DESCRIPTOR;

typedef struct _STAT_SERVER_0
{
    DWORD sts0_start;
    DWORD sts0_fopens;
    DWORD sts0_devopens;
    DWORD sts0_jobsqueued;
    DWORD sts0_sopens;
    DWORD sts0_stimedout;
    DWORD sts0_serrorout;
    DWORD sts0_pwerrors;
    DWORD sts0_permerrors;
    DWORD sts0_syserrors;
    DWORD sts0_bytessent_low;
    DWORD sts0_bytessent_high;
    DWORD sts0_bytesrcvd_low;
    DWORD sts0_bytesrcvd_high;
    DWORD sts0_avresponse;
    DWORD sts0_reqbufneed;
    DWORD sts0_bigbufneed;
} STAT_SERVER_0, *PSTAT_SERVER_0, *LPSTAT_SERVER_0;

typedef struct _TIME_OF_DAY_INFO
{
    DWORD tod_elapsedt;
    DWORD tod_msecs;
    DWORD tod_hours;
    DWORD tod_mins;
    DWORD tod_secs;
    DWORD tod_hunds;
    long tod_timezone;
    DWORD tod_tinterval;
    DWORD tod_day;
    DWORD tod_month;
    DWORD tod_year;
    DWORD tod_weekday;
} TIME_OF_DAY_INFO, *PTIME_OF_DAY_INFO, *LPTIME_OF_DAY_INFO;

typedef struct _NET_DFS_ENTRY_ID
{
    GUID Uid;
    [string] WCHAR * Prefix;
} NET_DFS_ENTRY_ID, *LPNET_DFS_ENTRY_ID;

typedef struct _NET_DFS_ENTRY_ID_CONTAINER
{
    unsigned long Count;
    [size_is(Count)] LPNET_DFS_ENTRY_ID Buffer;
} NET_DFS_ENTRY_ID_CONTAINER, *LPNET_DFS_ENTRY_ID_CONTAINER;

```

```

typedef struct _DFS_SITENAME_INFO
{
    unsigned long SiteFlags;
    [string,unique] WCHAR * SiteName;
} DFS_SITENAME_INFO, *PDFS_SITENAME_INFO, *LPDFS_SITENAME_INFO;

typedef struct _DFS_SITELIST_INFO
{
    unsigned long cSites;
    [size_is(cSites)] DFS_SITENAME_INFO Site[];
} DFS_SITELIST_INFO, *PDFS_SITELIST_INFO, *LPDFS_SITELIST_INFO;

typedef struct _SERVER_ALIAS_INFO_0 {
    [string] LMSTR     srvai0_alias;
    [string] LMSTR     srvai0_target;
    BOOLEAN    srvai0_default;
    ULONG      srvai0_reserved;
} SERVER_ALIAS_INFO_0, *PSERVER_ALIAS_INFO_0, *LPSERVER_ALIAS_INFO_0;

typedef struct _SERVER_ALIAS_INFO_0_CONTAINER {
    DWORD   EntriesRead;
    [size_is(EntriesRead)] LPSERVER_ALIAS_INFO_0 Buffer;
} SERVER_ALIAS_INFO_0_CONTAINER;

typedef struct _SERVER_ALIAS_ENUM_STRUCT {
    DWORD   Level;
    [switch_is(Level)] union _SERVER_ALIAS_ENUM_UNION {
        [case(0)]
        SERVER_ALIAS_INFO_0_CONTAINER *Level0;
        } ServerAliasInfo;
} SERVER_ALIAS_ENUM_STRUCT, *PSERVER_ALIAS_ENUM_STRUCT,
    *LPSERVER_ALIAS_ENUM_STRUCT;

typedef [switch_type(unsigned long)] union _SERVER_ALIAS_INFO
    { // for Get & Set Info
    [case(0)]
    LPSERVER_ALIAS_INFO_0 ServerAliasInfo0;
} SERVER_ALIAS_INFO, *PSERVER_ALIAS_INFO, *LPSERVER_ALIAS_INFO;

// This method not used on the wire
void Opnum0NotUsedOnWire(void);

// This method not used on the wire
void Opnum1NotUsedOnWire(void);

// This method not used on the wire
void Opnum2NotUsedOnWire(void);

// This method not used on the wire
void Opnum3NotUsedOnWire(void);

// This method not used on the wire
void Opnum4NotUsedOnWire(void);

// This method not used on the wire

```

```
void Opnum5NotUsedOnWire(void);

// This method not used on the wire
void Opnum6NotUsedOnWire(void);

// This method not used on the wire
void Opnum7NotUsedOnWire(void);

NET_API_STATUS
NetrConnectionEnum (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * Qualifier,
    [in,out] LPCONNECT_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferedMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);

NET_API_STATUS
NetrFileEnum (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * BasePath,
    [in,string,unique] WCHAR * UserName,
    [in,out] PFILE_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferedMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);

NET_API_STATUS
NetrFileGetInfo (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD FileId,
    [in] DWORD Level,
    [out, switch_is(Level)] LPFILE_INFO InfoStruct
);

NET_API_STATUS
NetrFileClose (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD FileId
);

NET_API_STATUS
NetrSessionEnum (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * ClientName,
    [in,string,unique] WCHAR * UserName,
    [in,out] PSESSION_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferedMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);

NET_API_STATUS
```

```
NetrSessionDel (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * ClientName,
    [in,string,unique] WCHAR * UserName
);

NET_API_STATUS
NetrShareAdd (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSHARE_INFO InfoStruct,
    [in,out,unique] DWORD * ParmErr
);

NET_API_STATUS
NetrShareEnum (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,out] LPSHARE_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);

NET_API_STATUS
NetrShareGetInfo (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * NetName,
    [in] DWORD Level,
    [out, switch_is(Level)] LPSHARE_INFO InfoStruct
);

NET_API_STATUS
NetrShareSetInfo (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * NetName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSHARE_INFO ShareInfo,
    [in,out,unique] DWORD * ParmErr
);

NET_API_STATUS
NetrShareDel (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * NetName,
    [in] DWORD Reserved
);

NET_API_STATUS
NetrShareDelSticky (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * NetName,
    [in] DWORD Reserved
);

NET_API_STATUS
```

```
NetrShareCheck (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * Device,
    [out] DWORD * Type
);

NET_API_STATUS
NetrServerGetInfo (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [out, switch_is(Level)] LPSERVER_INFO InfoStruct
);

NET_API_STATUS
NetrServerSetInfo (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPSERVER_INFO ServerInfo,
    [in,out,unique] DWORD * ParmErr
);

NET_API_STATUS
NetrServerDiskEnum (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in,out] DISK_ENUM_CONTAINER *DiskInfoStruct,
    [in] DWORD PreferedMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);

NET_API_STATUS
NetrServerStatisticsGet (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * Service,
    [in] DWORD Level,
    [in] DWORD Options,
    [out] LPSTAT_SERVER_0 *InfoStruct
);

NET_API_STATUS
NetrServerTransportAdd (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in] LPSERVER_TRANSPORT_INFO_0 Buffer
);

NET_API_STATUS
NetrServerTransportEnum (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,out] LPSERVER_XPORT_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferedMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);
```

```
NET_API_STATUS
NetrServerTransportDel (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in] LP SERVER_TRANSPORT_INFO_0 Buffer
);

NET_API_STATUS
NetrRemoteTOD (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [out] LPTIME_OF_DAY_INFO *BufferPtr
);

// This method not used on the wire
void Opnum29NotUsedOnWire(void);

NET_API_STATUS
NetprPathType(
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * PathName,
    [out] DWORD * PathType,
    [in] DWORD Flags
);

NET_API_STATUS
NetprPathCanonicalize(
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * PathName,
    [out,size_is(OutbufLen)] unsigned char * Outbuf,
    [in,range(0, 64000)] DWORD OutbufLen,
    [in,string] WCHAR * Prefix,
    [in,out] DWORD * PathType,
    [in] DWORD Flags
);

long
NetprPathCompare(
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * PathName1,
    [in,string] WCHAR * PathName2,
    [in] DWORD PathType,
    [in] DWORD Flags
);

NET_API_STATUS
NetprNameValidate(
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * Name,
    [in] DWORD NameType,
    [in] DWORD Flags
);

NET_API_STATUS
NetprNameCanonicalize(
```

```
[in,string,unique] SRVSVC_HANDLE ServerName,
[in,string] WCHAR * Name,
[out, size_is(OutbufLen)] WCHAR * Outbuf,
[in,range(0, 64000)] DWORD OutbufLen,
[in] DWORD NameType,
[in] DWORD Flags
);

long
NetprNameCompare(
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * Name1,
    [in,string] WCHAR * Name2,
    [in] DWORD NameType,
    [in] DWORD Flags
);

NET_API_STATUS
NetrShareEnumSticky (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,out] LPSHARE_ENUM_STRUCT InfoStruct,
    [in] DWORD PreferredMaximumLength,
    [out] DWORD * TotalEntries,
    [in,out,unique] DWORD * ResumeHandle
);

NET_API_STATUS
NetrShareDelStart (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * NetName,
    [in] DWORD Reserved,
    [out] PSHARE_DEL_HANDLE ContextHandle
);

NET_API_STATUS
NetrShareDelCommit (
    [in, out] PSHARE_DEL_HANDLE ContextHandle
);

DWORD
NetrpGetFileSecurity (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * ShareName,
    [in,string] WCHAR * lpFileName,
    [in] SECURITY_INFORMATION RequestedInformation,
    [out] PADT_SECURITY_DESCRIPTOR *SecurityDescriptor
);

DWORD
NetrpSetFileSecurity (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string,unique] WCHAR * ShareName,
    [in,string] WCHAR * lpFileName,
    [in] SECURITY_INFORMATION SecurityInformation,
    [in] PADT_SECURITY_DESCRIPTOR SecurityDescriptor
);
```

```
);

NET_API_STATUS
NetrServerTransportAddEx (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPTRANSPORT_INFO Buffer
);

// This method not used on the wire
void Opnum42NotUsedOnWire(void);

NET_API_STATUS
NetrDfsGetVersion(
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [out] DWORD * Version
);

NET_API_STATUS
NetrDfsCreateLocalPartition (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * ShareName,
    [in] GUID * EntryUid,
    [in,string] WCHAR * EntryPrefix,
    [in,string] WCHAR * ShortName,
    [in] LPNET_DFS_ENTRY_ID_CONTAINER RelationInfo,
    [in] int Force
);

NET_API_STATUS
NetrDfsDeleteLocalPartition (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] GUID * Uid,
    [in,string] WCHAR * Prefix
);

NET_API_STATUS
NetrDfsSetLocalVolumeState (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] GUID * Uid,
    [in,string] WCHAR * Prefix,
    [in] unsigned long State
);

// This method not used on the wire
void Opnum47NotUsedOnWire(void);

NET_API_STATUS
NetrDfsCreateExitPoint (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] GUID * Uid,
    [in,string] WCHAR * Prefix,
    [in] unsigned long Type,
    [in, range(0,32) ] DWORD ShortPrefixLen,
    [out, size_is(ShortPrefixLen)] WCHAR * ShortPrefix
```

```
);

NET_API_STATUS
NetrDfsDeleteExitPoint (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] GUID * Uid,
    [in,string] WCHAR * Prefix,
    [in] unsigned long Type
);

NET_API_STATUS
NetrDfsModifyPrefix (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] GUID * Uid,
    [in,string] WCHAR * Prefix
);

NET_API_STATUS
NetrDfsFixLocalVolume (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,string] WCHAR * VolumeName,
    [in] unsigned long EntryType,
    [in] unsigned long ServiceType,
    [in,string] WCHAR * StgId,
    [in] GUID * EntryUid,
    [in,string] WCHAR * EntryPrefix,
    [in] LPNET_DFS_ENTRY_ID_CONTAINER RelationInfo,
    [in] unsigned long CreateDisposition
);

NET_API_STATUS
NetrDfsManagerReportSiteInfo (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in,out,unique] LPDFS_SITELIST_INFO *ppSiteInfo
);

NET_API_STATUS
NetrServerTransportDelEx (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in, switch_is(Level)] LPTRANSPORT_INFO Buffer
);

NET_API_STATUS
NetrServerAliasAdd (
    [in,string,unique] SRVSVC_HANDLE          ServerName,
    [in]          DWORD             Level,
    [in, switch_is(Level)] LPSERVER_ALIAS_INFO InfoStruct
);

NET_API_STATUS
NetrServerAliasEnum (
    [in,string,unique] SRVSVC_HANDLE          ServerName,
    [in,out]          LPSERVER_ALIAS_ENUM_STRUCT InfoStruct,
    [in]          DWORD          PreferredMaximumLength,
```

```
[out]           LPDWORD      TotalEntries,
[in,out,unique] LPDWORD      ResumeHandle
);

NET_API_STATUS
NetrServerAliasDel (
    [in,string,unique] SRVSVC_HANDLE      ServerName,
    [in]             DWORD          Level,
    [in, switch_is(Level)] LPSERVER_ALIAS_INFO   InfoStruct
);

NET_API_STATUS
NetrShareDelEx (
    [in,string,unique] SRVSVC_HANDLE ServerName,
    [in] DWORD Level,
    [in,switch_is(Level)] LPSHARE_INFO ShareInfo
);
}
```

# 7 Appendix B: Product Behavior

Article10/04/2021

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Windows NT operating system
- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system
- Windows Server 2022 operating system
- Windows 11 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise

specified. If a product edition appears with the product version, behavior is different in that product edition.

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

<1> [Section 1.8](#): Windows uses only the values in [\[MS-EERR\]](#).

<2> [Section 2.1](#): Windows uses the identity of the caller to perform method-specific access checks.

<3> [Section 2.2.2.1](#): Windows-based [SMB clients](#) set this field based on the version and service pack level of the Windows operating system. Windows Vista operating system and later, windows\_server\_2008 operating system and later, set this field to an empty string. The following table specifies the Sessionclient string and corresponding Windows operating system version.

Sessionclient String	Windows Operating System Version
"Administration Tools Pack"	Windows Server 2003 operating system with Service Pack 1 (SP1)
"Windows 2002 Service Pack 2"	Windows XP operating system Service Pack 2 (SP2)
"Windows 5.0"	Windows 2000
"Windows NT 1381"	Windows NT 4.0 operating system
"Windows 4.0"	Windows 98 operating system and Windows 98 operating system Second Edition
"DOS LM 1.0"	LAN Manager for MS-DOS 1.0 clients
"DOS LM 2.0"	LAN Manager for MS-DOS 2.0 clients
"OS/2 LM 1.0"	LAN Manager for MS-OS/2 1.0 clients
"OS/2 LM 2.0"	LAN Manager for MS-OS/2 2.0 clients

<4> [Section 2.2.2.1](#): Windows-based servers currently do not enforce any limits on the **Sessionclient** string size and will accept any string containing 0 or more characters. The existing Windows clients limit the size to less than 256 bytes.

<5> [Section 2.2.2.6](#): Use PLATFORM\_ID\_NT for Windows NT Server operating system operating system and later, Windows 2000 operating system and later.

<6> Section 2.2.2.6: Windows clients treat any **PlatformID** values not specified in the table as unknown platforms.

<7> Section 2.2.2.13: Entry refers to a Windows NT, Windows 2000, or Windows XP server.

<8> Section 2.2.3.7: The **ServerInfo103** parameter and **SERVER\_INFO\_103** structure are applicable to Windows Server 2008 R2 operating system operating system and later.

<9> Section 2.2.4.13: Windows-based SMB clients set this field based on the version and service pack level of the Windows operating system. Windows Vista operating system and later, Windows Server 2008 operating system and later, set this field to an empty string. The following table specifies the Sessionclient string and corresponding Windows version.

Sessionclient String	Windows Operating System Version
"Administration Tools Pack"	Windows Server 2003 with SP1
"Windows 2002 Service Pack 2"	Windows XP SP2
"Windows 5.0"	Windows 2000
"Windows NT 1381"	Windows NT 4.0
"Windows 4.0"	Windows 98 and Windows 98 Second Edition
"DOS LM 1.0"	LAN Manager for MS-DOS 1.0 clients
"DOS LM 2.0"	LAN Manager for MS-DOS 2.0 clients
"OS/2 LM 1.0"	LAN Manager for MS-OS/2 1.0 clients
"OS/2 LM 2.0"	LAN Manager for MS-OS/2 2.0 clients

<10> Section 2.2.4.15: Windows-based SMB clients set this field based on the version and service pack level of the Windows operating system. Windows Vista operating system and later, Windows Server 2008 operating system and later set this field to an empty string. The following table specifies the Sessionclient string and corresponding Windows operating system version.

Sessionclient String	Windows Operating System Version
"Administration Tools Pack"	Windows Server 2003 with SP1
"Windows 2002 Service Pack 2"	Windows XP SP2
"Windows 5.0"	Windows 2000

Sessionclient String	Windows Operating System Version
"Windows NT 1381"	Windows NT 4.0
"Windows 4.0"	Windows 98 and Windows 98 Second Edition
"DOS LM 1.0"	LAN Manager for MS-DOS 1.0 clients
"DOS LM 2.0"	LAN Manager for MS-DOS 2.0 clients
"OS/2 LM 1.0"	LAN Manager for MS-OS/2 1.0 clients
"OS/2 LM 2.0"	LAN Manager for MS-OS/2 2.0 clients

<11> [Section 2.2.4.29](#): SHI1005\_FLAGS\_ACCESS\_BASED\_DIRECTORY\_ENUM is supported only on servers running Windows Server 2003 with SP1, Windows Server 2008 operating system and later, Windows 7 operating system and later.

<12> [Section 2.2.4.29](#): SHI1005\_FLAGS\_FORCE\_LEVELII\_OPLOCK is supported on Windows Server 2008 R2 operating system and later.

<13> [Section 2.2.4.29](#): SHI1005\_FLAGS\_ENABLE\_HASH is supported on Windows Server 2008 R2 operating system and later.

<14> [Section 2.2.4.29](#): SHI1005\_FLAGS\_ENABLE\_CA is supported on Windows Server 2012 operating system and later.

<15> [Section 2.2.4.29](#): SHI1005\_FLAGS\_ENCRYPT\_DATA is supported on Windows 8 operating system and later, Windows Server 2012 operating system and later.

<16> [Section 2.2.4.31](#): SHARE\_INFO\_1501\_I is supported after Windows 2000.

<17> [Section 2.2.4.43](#): The following values are returned by Windows-based servers for different versions of the Windows operating system.

Operating system	Major version
Windows NT 4.0	4
Windows 2000	5
Windows XP	5
Windows Server 2003	5
Windows Vista	6
Windows Server 2008	6

Operating system	Major version
Windows 7	6
Windows Server 2008 R2	6
Windows 8	6
Windows Server 2012	6
Windows 8.1	6
Windows Server 2012 R2	6
Windows 10	10
Windows Server 2016	10
Windows Server operating system	10
Windows Server 2019	10
Windows 10 v2004 operating system	10
Windows Server 2022	10
Windows 11	10

<18> Section 2.2.4.43: The following values are returned by Windows-based servers for different versions of the Windows operating system.

Operating system	Minor version
Windows NT 4.0	0
Windows 2000	0
Windows XP	1
Windows Server 2003	2
Windows Vista	0
Windows Server 2008	0
Windows 7	1
Windows Server 2008 R2	1
Windows 8	2
Windows Server 2012	2

Operating system	Minor version
Windows 8.1	3
Windows Server 2012 R2	3
Windows 10	0
Windows Server 2016	0
Windows Server operating system	0
Windows Server 2019	0
Windows 10 v2004	0
Windows Server 2022	0
Windows 11	0

<19> [Section 2.2.4.43](#): SRV\_HASH\_GENERATION\_ACTIVE is enabled only if SRV\_SUPPORT\_HASH\_GENERATION is enabled.

<20> [Section 2.2.4.46](#): The allowed range of values on Windows NT 4.0 is 1 to 2,048, inclusive.

<21> [Section 2.2.4.46](#): The allowed range of values for get operations on Windows NT 4.0 and Windows 2000 is 512 to 65,535, inclusive.

<22> [Section 2.2.4.46](#): The allowed range of values for get operations on Windows NT 4.0 is 1 to 20, inclusive.

<23> [Section 2.2.4.46](#): The allowed range of values in Windows is from 0x00100000 to 0xFFFFFFFF, inclusive.

<24> [Section 2.2.4.46](#): The allowed range of values in Windows is from 0x00100000 to 0xFFFFFFFF, inclusive.

<25> [Section 2.2.4.46](#): The allowed range of values for Windows NT 4.0, Windows 2000, and Windows XP is 2 to 32, inclusive.

<26> [Section 2.2.4.46](#): The allowed range of values for Windows NT 4.0, Windows 2000, and Windows XP is 2 to 100, inclusive.

<27> [Section 2.2.4.96](#): Following are examples of values that this field can have for Microsoft-supported protocols:

- NETBT (NetBIOS over TCP/IP)

On Windows 2000 operating system and later, Windows Server 2003 operating system and later, the format is as follows, where the value between braces is the [GUID](#) of the underlying physical interface that is generated by the operating system at installation time: \Device\NetBT\_Tcpip\_{2C9725F4-151A-11D3-AEC-C3B211BD350B}

On Windows NT 4.0, the format is as follows, where DC21X41 is the name for the adapter chosen by the manufacturer: \Device\NetBT\_DC21X41

- Direct hosting of SMB over TCP/IP (NetBIOS-less SMB)

This protocol is available only on Windows 2000 operating system and later, Windows Server 2003 operating system and later. The format is:  
\Device\NetbiosSmb

- NwInk (the Microsoft version of the Novell IPX/SPX Protocol [\[NWLINK\]](#))

This protocol is not installed by default. It provides the following two transports:  
\Device\NwInklpx and \Device\NwInkNb

- NetBEUI

This protocol is supported only on Windows 2000 and Windows NT 4.0. The value between braces is the GUID of the underlying physical port generated by the operating system at installation time. The NdisWanNbfOut/NdisWanNbfln devices correspond to bindings between the NetBEUI transport and NDISWAN driver. The format options are:

\Device\Nbf\_{868B258E-252B-4F65-A383-18803360701F}

\Device\Nbf\_NdisWanNbfOut{77C17309-B558-4096-8A2B-2D1E9E4FC932}

\Device\Nbf\_NdisWanNbfln{331BB986-F9B0-406C-9FA2-36425F52CC05}

[<28> Section 2.2.4.96](#): This member is usually the NetBIOS name that the server is using, or it can represent an SMB/IPX name.

[<29> Section 2.2.4.96](#): The server normalizes this to 16 characters by truncating the given length to this value if it is larger, or padding the transport address buffer with the blank character (0x20) until the length is 16.

[<30> Section 2.2.4.96](#): Following are examples of values this field can have for Microsoft-supported protocols:

- NETBT (NetBIOS over TCP/IP)

The MAC address of the n/w device, for example: 00065b5da43f

- NetBIOS over SMB

000000000000

- NwInk (the Microsoft version of the Novell IPX/SPX Protocol [NWLINK])

The MAC address of the n/w device, for example: 00065b5da43f

- NetBEUI

The MAC address of the n/w device for the non-NdisWan devices, for example:  
00065b5da43f

For the NdisWan devices, this pointer is an index into internal connection tables of the driver. The first two characters are generated randomly by using the current system tick count and the next two by using the current system time at installation. The last eight characters are always 20524153 and stand for the string "RAS" including the leading blank. For example: d2e820524153.

[\*\*<31> Section 3.1.1:\*\*](#) In Windows, virtual shares are implemented in **DFS**, which is a referral service to SMB shares, as specified in [\[MS-DFSC\]](#). The DFS abstract model is specified in [\[MS-DFSC\]](#). DFS is a special type of share that is relevant to the Windows client.

[\*\*<32> Section 3.1.1:\*\*](#) By default, Windows-based SMB and SMB2 servers are configured to listen on both Direct TCP as specified in [\[MS-SMB\]](#) sections [1.9](#) and [2.1](#), and NetBIOS over TCP as specified in [\[MS-CIFS\]](#) section [2.1.1.2](#). Windows-based CIFS servers are configured to listen on additional NetBIOS-based transports as specified in [\[MS-CIFS\]](#) section [2.1](#), when the appropriate link layers are available. These settings can also be obtained via policy or DHCP configuration.

[\*\*<33> Section 3.1.1:\*\*](#) Windows-specific transport names are as specified in the product behavior note for svti3\_transportname in section [2.2.4.96](#).

[\*\*<34> Section 3.1.1:\*\*](#) Windows stores the list of all active shares that are identified by a share identifier in the registry, at the path  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver.

[\*\*<35> Section 3.1.1.7:\*\*](#) This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

[\*\*<36> Section 3.1.3:\*\*](#) Windows-based servers set this flag to SV\_TYPE\_NT.

<37> Section 3.1.3: Windows stores these named pipes in the registry at the path "\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\NullSessionPipes".

The following table lists the named pipe names that an anonymous user is allowed to open. The default behavior of Windows in allowing anonymous access to certain pipes has become more restrictive over time.

Operating system	Pipes
Windows NT 4.0	comnap, commode, sql\query, spoolss, epmapper, locator, lsarpc, samr, netlogon, wkssvc, svrsvc, and browser
Windows 2000	comnap, commode, sql\query, spoolss, epmapper, locator, trkwks, trksrv, lsarpc, samr, netlogon, wkssvc, svrsvc, and browser
Windows XP	comnap, commode, sql\query, spoolss, browser
Windows Server 2003	comnap, commode, sql\query, spoolss, netlogon, lsarpc, samr, browser
Windows Vista, Windows Server 2008	browser
Windows Server 2008 R2 operating system and later, Windows 7 operating system and later	No pipes are allowing anonymous access

<38> Section 3.1.3: In Windows, the dependency chain for a service group ensures that the server service starts before the SMB and SMB2 services.

<39> Section 3.1.3: By default, Windows sets the values as follows:

- sv103\_version\_major is set to 3.
- sv103\_version\_minor is set to 10.
- sv103\_comment is set to empty string.
- sv103\_users is set to 0xFFFFFFFF.
- sv103\_disc is set to 15.
- sv103\_hidden is set to FALSE.
- sv103\_announce is set to 240.
- sv103\_anndelta is set to 3000.

<40> Section 3.1.3: By default, Windows sets the values as follows:

- sv599\_sessopens is set to 2048.
- sv599\_sessvcs is set to 1.
- sv599\_opensearch is set to 2048.
- sv599\_sizreqbuf is set to 4356.
- sv103\_disc is set to 15.
- sv599\_initworkitems is set to 4.
- sv599\_maxworkitems is set to 16.
- sv599\_rawworkitems is set to 4.
- sv599\_irpstacksize is set to 11.
- sv599\_maxrawbuflen is set to 65535.
- sv599\_sessusers is set to 2048.
- sv599\_sessconns is set to 2048.
- sv599\_maxpagedmemoryusage is set to 0xFFFFFFFF.
- sv599\_maxnonpagedmemoryusage is set to 0xFFFFFFFF.
- sv599\_enablessoftcompat is set to TRUE.
- sv599\_enableforcedlogoff is set to TRUE.
- sv599\_timesource is set to FALSE.
- sv599\_acceptdownlevelapis is set to TRUE.
- sv599\_lmannounce is set to FALSE.
- sv599\_domain is set to "DOMAIN".
- sv599\_maxcopyreadlen is set to 8192.
- sv599\_maxcopywritelen is set to 0.
- sv599\_minkeepsearch is set to 480.
- sv599\_maxkeepsearch is set to 3600.

- sv599\_minkeepcomplesearch is set to 240.
- sv599\_maxkeepcomplesearch is set to 600.
- sv599\_threadcountadd is set to 2.
- sv599\_numblockthreads is set to 2.
- sv599\_scavtimeout is set to 30.
- sv599\_minrcvqueue is set to 2.
- sv599\_minfreeworkitems is set to 2.
- sv599\_xactmemsize is set to 0x100000.
- sv599\_threadpriority is set to 1.
- sv599\_maxmpxct is set to 50.
- sv599\_oplockbreakwait is set to 35.
- sv599\_oplockbreakresponsewait is set to 35.
- sv599\_enableoplocks is set to TRUE.
- sv599\_enableoplockforceclose is set to FALSE.
- sv599\_enablefcbopens is set to TRUE.
- sv599\_enableraw is set to TRUE.
- sv599\_enablesharednetdrives is set to FALSE.
- sv599\_minfreeconnections is set to 2.
- sv599\_maxfreeconnections is set to 2.
- sv599\_initsesstable is set to 4.
- sv599\_initconntable is set to 8.
- sv599\_initfiletable is set to 16.
- sv599\_initsearchtable is set to 8.
- sv599\_alertschedule is set to 5.
- sv599\_errorthreshold is set to 10.

- sv599\_networkerrorthreshold is set to 5.
- sv599\_diskspacethreshold is set to 10.
- sv599\_maxlinkdelay is set to 60.
- sv599\_minlinkthroughput is set to 0.
- sv599\_linkinfovalidtime is set to 60.
- sv599\_scavqosinfoupdatetime is set to 300.
- sv599\_maxworkitemidletime is set to 30.

**<41> Section 3.1.4:** In Windows Server 2003 operating system and later, messages that are discussed in section NetrDfsGetVersion (Opnum 43) (section 3.1.4.35) through section NetrDfsManagerReportSiteInfo (Opnum 52) (section 3.1.4.43) (that is, all messages whose names begin with NetrDfs) have been deprecated. Calling them on Windows Server 2003 operating system and later returns an implementation-specific error code.

**<42> Section 3.1.4:** Windows implementation uses the [RPC](#) protocol to retrieve the identity of the caller specified in [\[MS-RPCE\]](#) section 3.3.3.4.3. The server uses the underlying Windows security subsystem to determine the permissions for the caller. If the caller does not have the required permissions to execute a specific method, the method call fails with an implementation-specific error code.

**<43> Section 3.1.4.1:** The Windows implementation checks to see whether the caller is a member of the Administrator, Server or Print Operator, or Power User local group.

**<44> Section 3.1.4.1:** If the caller is not a member of the Administrator, Server or Print Operator, or Power User local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

**<45> Section 3.1.4.2:** The Windows implementation checks to see whether the caller is a member of the Administrator or Server Operator local group.

**<46> Section 3.1.4.2:** If the caller is not a member of the Administrator or Server Operator local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

**<47> Section 3.1.4.3:** The Windows implementation checks to see whether the caller is a member of the Administrator or Server Operator local group.

**<48> Section 3.1.4.3:** If the caller is not a member of the Administrator or Server Operator local group, Windows-based servers fail the call with the error code

ERROR\_ACCESS\_DENIED.

<49> [Section 3.1.4.4](#): The Windows implementation checks to see whether the caller is a member of the Administrator or Server Operator local group.

<50> [Section 3.1.4.4](#): If the caller is not a member of the Administrator or Server Operator local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<51> [Section 3.1.4.5](#): The Windows implementation checks to see whether the caller is a member of the Administrator or Server Operator local group.

<52> [Section 3.1.4.5](#): If the caller is not a member of the Administrator or Server Operator local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<53> [Section 3.1.4.6](#): The Windows implementation checks to see whether the caller is a member of the Administrators or Server Operators local group.

<54> [Section 3.1.4.6](#): If the caller is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<55> [Section 3.1.4.7](#): If the requested share is a file share, the Windows implementation checks whether the caller is a member of the Administrators, System Operators, or Power Users local group. If the requested share is a printer share, the Windows implementation checks whether the caller is a member of the Print Operators group.

<56> [Section 3.1.4.7](#): Only members of the Administrators, System Operators, or Power Users local group can add file shares with a call to the [NetrShareAdd](#) method. A member of the Print Operators group can add printer shares. If this condition is not met, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<57> [Section 3.1.4.8](#): The Windows implementation checks to see whether the caller is a member of the Administrator or Server Operator local group.

<58> [Section 3.1.4.8](#): If the caller is not a member of the Administrator or Server Operator local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<59> [Section 3.1.4.9](#): The server stores information about [sticky shares](#) in the Windows registry.

<60> [Section 3.1.4.10](#): If the requested level is 2, 502, or 503, the Windows implementation checks to see whether the caller is in the Administrators, Server or Print

Operators, or Power Users local group. No special group membership is required for other levels.

<61> [Section 3.1.4.10](#): Only members of the Administrators, Server or Print Operators, or Power Users local group can successfully execute the **NetrShareGetInfo** message at levels 2, 502, or 503. No special group membership is required for the other levels. If this condition is not met, Windows-based servers fail the call with the error code **ERROR\_ACCESS\_DENIED**.

<62> [Section 3.1.4.11](#): If the value of *Level* is 1005, the *shi1005\_flags* parameter contains **SHI1005\_FLAGS\_ENABLE\_HASH**, and the server does not support branch cache, the server fails the call with the error code **ERROR\_NOT\_SUPPORTED**. This error is supported in Windows Server 2008 R2 operating system and later.

<63> [Section 3.1.4.11](#): If the value of *Level* is 1005, the *shi1005\_flags* parameter contains **SHI1005\_FLAGS\_ENABLE\_HASH**, and the server does not install the branch cache component, the server fails the call with the error code **ERROR\_SERVICE\_DOES\_NOT\_EXIST**. This error is supported in Windows Server 2008 R2 operating system and later.

<64> [Section 3.1.4.11](#): If *Level*=1005 and *shi\*\_type* do not have the flag **STYPE\_DISKTREE**, the server fails the call by using an implementation-specific error code.

<65> [Section 3.1.4.11](#): Windows checks whether the caller is a member of the Administrators or Server Operators local group.

<66> [Section 3.1.4.11](#): If the caller is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code **ERROR\_ACCESS\_DENIED**.

<67> [Section 3.1.4.12](#): Windows uses the registry as permanent storage.

<68> [Section 3.1.4.12](#): Windows-based clients set this field to an arbitrary value. The actual value does not affect server behavior because the server is required to ignore this field.

<69> [Section 3.1.4.12](#): If the specified share is a file share, the Windows implementation checks to see whether the caller is a member of the Administrators, Server Operators, or Power Users local group. If the specified share is a printer share, the Windows implementation checks to see whether the caller is a member of the Print Operator group.

<70> [Section 3.1.4.12](#): Only members of the Administrators, Server Operators, or Power Users local group can successfully delete file shares by using a [NetrShareDel](#) message call. The Print Operator can delete printer shares. If the caller does not meet these requirements, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<71> [Section 3.1.4.13](#): Windows-based clients set this field to an arbitrary value. The actual value does not affect server behavior because the server is required to ignore this field.

<72> [Section 3.1.4.13](#): Windows uses the registry as the permanent storage.

<73> [Section 3.1.4.13](#): If the specified share is a file share, the Windows implementation checks to see whether the caller is a member of the Administrators, Server Operators, or Power Users local group. If the specified share is a printer share, the Windows implementation checks to see whether the caller is a member of the Print Operator group.

<74> [Section 3.1.4.13](#): Only members of the Administrators, Server Operators, or Power Users local group can successfully delete file shares with a [NetrShareDelSticky](#) message call. The Print Operator can delete printer shares. If the caller does not meet these requirements, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<75> [Section 3.1.4.14](#): If the specified share is a file share, the Windows implementation checks to see whether the caller is a member of the Administrators, Server Operators, or Power Users local group. If the share that is specified is a printer share, the Windows implementation checks to see whether the caller is a member of the Print Operator group.

<76> [Section 3.1.4.14](#): Only members of the Administrators, Server Operators, or Power Users local group can successfully delete file shares with a [NetrShareDelStart](#) message call. The Print Operator can delete printer shares. If the caller does not meet these requirements, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<77> [Section 3.1.4.17](#): The value 103 is supported in Windows Server 2008 R2 operating system and later.

<78> [Section 3.1.4.17](#): The SERVER\_INFO\_103 structure is supported in Windows Server 2008 R2 operating system and later.

<79> [Section 3.1.4.17](#): If the level is 503, the Windows implementation checks whether the caller is a member of the Administrators or Server Operators local group. If the level

is 102 or 502, the Windows implementation checks whether the caller is a member of one of the groups previously mentioned or is a member of the Power Users local group.

<80> [Section 3.1.4.17](#): If the caller is not a member of the Administrators or Server Operators local group and the level is 503, the server fails the call with an implementation-specific error code. If the caller is not a member of one of the groups previously mentioned, the caller is not a member of the Power Users local group, and the level is 102 or 502, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<81> [Section 3.1.4.18](#): This information is stored in the Windows registry.

<82> [Section 3.1.4.18](#): If any member of the structure *ServerInfo* is found invalid, the server fails the call with an implementation-specific error code.

<83> [Section 3.1.4.18](#): The Windows implementation checks whether the client is a member of the Administrators or Server Operators local group.

<84> [Section 3.1.4.18](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<85> [Section 3.1.4.19](#): The Windows implementation checks to see whether the client is a member of the Administrators or Server Operators local group.

<86> [Section 3.1.4.19](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<87> [Section 3.1.4.20](#): The Windows implementation checks to see whether the client is a member of the Administrators or Server Operators local group.

<88> [Section 3.1.4.20](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<89> [Section 3.1.4.21](#): No special group membership is required to successfully execute this message.

<90> [Section 3.1.4.21](#): No special group membership is required to successfully execute this message.

<91> [Section 3.1.4.22](#): The Windows implementation checks to see if the client is a member of the Administrators or Server Operators local group.

<92> [Section 3.1.4.22](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<93> [Section 3.1.4.23](#): The Windows implementation checks to see whether the client is a member of the Administrators or Server Operators local group.

<94> [Section 3.1.4.23](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<95> [Section 3.1.4.24](#): The Windows implementation checks to see whether the caller is a member of the Administrators, Server Operators, or Power Users local group.

<96> [Section 3.1.4.24](#): If the caller is not a member of the Administrators, Server Operators, or Power Users local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<97> [Section 3.1.4.25](#): The Windows implementation checks to see if the client is a member of the Administrators or Server Operators local group.

<98> [Section 3.1.4.25](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<99> [Section 3.1.4.26](#): Windows Vista operating system and later, Windows Server 2008 operating system and later return 0x00000000 even when the transport that is being deleted does not exist or has already been deleted.

<100> [Section 3.1.4.26](#): The method `NetrServerTransportDelEx` is defined only on Windows XP operating system and later, Windows Vista operating system and later.

<101> [Section 3.1.4.26](#): The Windows implementation checks to see whether the client is a member of the Administrators or Server Operators local group.

<102> [Section 3.1.4.26](#): If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<103> [Section 3.1.4.27](#): Windows-based servers fail the call with an ERROR\_INVALID\_PARAMETER if the file does not exist.

<104> [Section 3.1.4.27](#): In order to read the owner, group, or discretionary access control list (DACL) [MS-DTYP] from the security descriptor for the specified file or directory or the DACL for the file or directory, the caller has to have READ\_CONTROL

access, or the caller has to be the owner of the file or directory. In order to read the system access control list (SACL) [MS-DTYP] of a file or directory, the SE\_SECURITY\_NAME privilege [MS-DTYP] has to be enabled for the calling process.

<105> [Section 3.1.4.27](#): If the caller does not meet the security measures that are specified for the Windows implementation, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<106> [Section 3.1.4.28](#): This message executes successfully only if the following conditions are met:

- If the owner of the object is being set, the client has to either have WRITE\_OWNER permission or be the owner of the object.
- If the DACL of the object is being set, the client has to either have WRITE\_DAC permission or be the owner of the object.
- If the SACL of the object is being set, the SE\_SECURITY\_NAME privilege has to be enabled for the client.

<107> [Section 3.1.4.28](#): If the server does not meet the security measures that are specified for the Windows implementation, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<108> [Section 3.1.4.29](#): No security restrictions are imposed by Windows implementations on the caller.

<109> [Section 3.1.4.29](#): No security restrictions are imposed by Windows implementations on the caller.

<110> [Section 3.1.4.30](#): Windows-based servers fail the call with an ERROR\_INVALID\_PARAMETER error code if the value of Flags is other than 0x00000000, 0x00000001, 0x80000000, or 0x80000001.

<111> [Section 3.1.4.30](#): Windows uses "\\" as the path separator.

<112> [Section 3.1.4.30](#): Windows uses "\\" as the path separator. The Windows implementation does the following during canonicalization:

- All macros in the input file name (\., .\., \..) are removed and replaced by path components.
- Any required translations are performed on the path specification:
  - UNIX-style "/" converted to DOS-style "\\"

- Specific transliteration

**Note** The input case is NOT converted. The underlying file system can be case insensitive. The path is passed through, with the case exactly as presented by the caller.

- Device names (that is, namespace controlled by the server) are canonicalized by converting device names to uppercase and removing trailing colons in all but disk devices.

<113> [Section 3.1.4.30](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<114> [Section 3.1.4.30](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<115> [Section 3.1.4.31](#): If the *Flags* parameter is 1, the server ignores the *PathType* parameter.

<116> [Section 3.1.4.31](#): The server does a standard C string comparison on the canonicalized path names and returns the result.

<117> [Section 3.1.4.31](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<118> [Section 3.1.4.31](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<119> [Section 3.1.4.32](#): Windows-based servers fail the call with ERROR\_INVALID\_PARAMETER if the value of Flags is other than 0x00000000 and 0x80000000.

<120> [Section 3.1.4.32](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<121> [Section 3.1.4.32](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<122> [Section 3.1.4.33](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<123> [Section 3.1.4.33](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<124> [Section 3.1.4.34](#): The server does a string comparison and returns the results for all NameTypes except NAMETYPE\_COMPUTER, NAMETYPE\_WORKGROUP, and

NAMETYPE\_DOMAIN. For these, the server first converts the names to the corresponding OEM character set for the local environment and then does a string comparison on the resultant strings.

<125> [Section 3.1.4.34](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<126> [Section 3.1.4.34](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<127> [Section 3.1.4.35](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<128> [Section 3.1.4.35](#): This method is supported only in Windows 2000 and Windows XP. Otherwise, it returns an ERROR\_FILE\_NOT\_FOUND error code.

<129> [Section 3.1.4.35](#): The server always sets the *Version* parameter to zero.

<130> [Section 3.1.4.35](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<131> [Section 3.1.4.35](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<132> [Section 3.1.4.36](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<133> [Section 3.1.4.36](#): Windows implementations use the CoCreateGuid() API to create a unique GUID. For more information about the CoCreateGuid() API, see [\[MSDN-CoCreateGuid\]](#).

<134> [Section 3.1.4.36](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<135> [Section 3.1.4.36](#): Both *ShortName* and *EntryPrefix* are used to match a DFS path. If the latter does not match but the first matches, the server tries to use that.

<136> [Section 3.1.4.36](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<137> [Section 3.1.4.36](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<138> [Section 3.1.4.37](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<139> [Section 3.1.4.37](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<140> [Section 3.1.4.37](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<141> [Section 3.1.4.37](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<142> [Section 3.1.4.38](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<143> [Section 3.1.4.38](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<144> [Section 3.1.4.38](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<145> [Section 3.1.4.38](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<146> [Section 3.1.4.39](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<147> [Section 3.1.4.39](#): The *ShortPrefix* parameter is only supported in Windows 2000 and Windows XP. When supported, *ShortPrefix* has one leading backslash instead of the usual two, and is without a terminating null character. If the *ShortPrefix* size is greater than the size specified in *ShortPrefixLen*, it returns a NULL (zero-length) string and does not fail. Otherwise, it returns ERROR\_NOT\_SUPPORTED.

<148> [Section 3.1.4.39](#): This method is supported only in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<149> [Section 3.1.4.39](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<150> [Section 3.1.4.39](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<151> [Section 3.1.4.40](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<152> [Section 3.1.4.40](#): This method is supported only in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<153> [Section 3.1.4.40](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<154> [Section 3.1.4.40](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<155> [Section 3.1.4.41](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<156> [Section 3.1.4.41](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<157> [Section 3.1.4.41](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<158> [Section 3.1.4.41](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<159> [Section 3.1.4.42](#): This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<160> [Section 3.1.4.42](#): The target is specified in the form of a Windows NT path name. Windows subsystem DLLs add the prefix "\??" to names that are passed by Windows applications that reference objects in \DosDevices. "\DosDevices" represents a symbolic link to a directory in the object manager namespace that stores MS-DOS device names as \DosDevices\DosDeviceName. An example of a device with an MS-DOS device name is the serial port, COM1. It has the MS-DOS device name \DosDevices\COM1. Likewise, the C: drive has the name \DosDevices\C:.

<161> [Section 3.1.4.42](#): This method is supported only in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

<162> [Section 3.1.4.42](#): Windows subsystem DLLs add the prefix "\??" to names that are passed by Windows applications that reference objects in \DosDevices. "\DosDevices" represents a symbolic link to a directory in the object manager namespace that stores MS-DOS device names.

<163> [Section 3.1.4.42](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<164> [Section 3.1.4.42](#): No security restrictions are imposed by Windows-based server implementations on the caller.

<165> [Section 3.1.4.43](#): Windows allows the server administrator to configure a static list of site names to be returned by this method. If the Active Directory administrator

changes [site](#) names and the server administrator does not update the static list, or the server administrator makes an error, this method will return names that are not current Active Directory site names.

[\*\*<166> Section 3.1.4.43:\*\*](#) This method is only supported in Windows 2000 and Windows XP. Otherwise, it returns an implementation-specific error code.

[\*\*<167> Section 3.1.4.43:\*\*](#) Windows implementations first seek within the registry subkey `SYSTEM\CurrentControlSet\Services\DfsDriver\CoveredSites` for a value that matches the *ServerName* parameter. If that value is present and a REG\_MULTI\_SZ value, its contents form the list returned by the method. Otherwise, the list is formed in the next two steps.

First, the implementation makes a call to the local Netlogon Remote Protocol server using the `DsrGetSiteName` method, as specified in [\[MS-NRPC\]](#) section [3.5.4.3.6](#). In this call, a NULL ComputerName argument is provided. If successful and a site name is returned, this name forms part of the response. This site name will be marked with the `DFS_SITE_PRIMARY` flag, as specified in section [2.2.4.109](#) of this document.

Second, the implementation seeks the registry value `SYSTEM\CurrentControlSet\Services\DfsDriver\CoveredSites\CoveredSites`. If that value is present and a REG\_MULTI\_SZ value, its contents form the rest of the list returned by the method.

[\*\*<168> Section 3.1.4.43:\*\*](#) No security restrictions are imposed by Windows-based server implementations on the caller.

[\*\*<169> Section 3.1.4.43:\*\*](#) No security restrictions are imposed by Windows-based server implementations on the caller.

[\*\*<170> Section 3.1.4.44:\*\*](#) The Windows implementation checks to see if the client is a member of the Administrators or Server Operators local group.

[\*\*<171> Section 3.1.4.44:\*\*](#) If the client is not a member of the Administrators or Server Operators local group, Windows-based servers fail the call with the error code `ERROR_ACCESS_DENIED`.

[\*\*<172> Section 3.1.4.45:\*\*](#) The Windows implementation checks to see if the caller is a member of the Administrator or Server Operator local group.

[\*\*<173> Section 3.1.4.45:\*\*](#) If the caller is not a member of the Administrator or Server Operator local group, Windows-based servers fail the call with the error code `ERROR_ACCESS_DENIED`.

<174> [Section 3.1.4.46](#): If the specified share is a file share, the Windows implementation checks to see whether the caller is a member of the Administrators, Server Operators, or Power Users local group. If the specified share is a printer share, the Windows implementation checks to see whether the caller is a member of the Print Operator group.

<175> [Section 3.1.4.46](#): Only members of the Administrators, Server Operators, or Power Users local group can successfully delete file shares by using a NetrShareDel message call. The Print Operator can delete printer shares. If the caller does not meet these requirements, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<176> [Section 3.1.4.47](#): Windows uses the registry as permanent storage.

<177> [Section 3.1.4.47](#): If the specified share is a file share, the Windows implementation checks to see whether the caller is a member of the Administrators, Server Operators, or Power Users local group. If the specified share is a printer share, the Windows implementation checks to see whether the caller is a member of the Print Operator group.

<178> [Section 3.1.4.47](#): Only members of the Administrators, Server Operators, or Power Users local group can successfully delete file shares by using a NetrShareDel message call. The Print Operator can delete printer shares. If the caller does not meet these requirements, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

<179> [Section 3.1.6.13](#): The Windows implementation checks to see whether the caller is a member of the Administrator, Server Operator, or Power User local group.

<180> [Section 3.1.6.13](#): If the caller is not a member of the Administrator, Server Operator, or Power User local group, Windows-based servers fail the call with the error code ERROR\_ACCESS\_DENIED.

# 8 Change Tracking

Article10/04/2021

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

Section	Description	Revision class
<a href="#">2.2.4.46 SERVER_INFO_599</a>	11226 : Removed THREAD_BASE_PRIORITY_LOWRT.	Minor
<a href="#">3.1.4 Message Processing Events and Sequencing Rules</a>	11226 : Replaced ERROR_NOT_IMPLEMENTED with ERROR_NOT_SUPPORTED.	Minor
<a href="#">3.1.4.1 NetrConnectionEnum (Opnum 8)</a>	11227 : Updated the processing rules for CONNECTION_INFO_0_CONTAINER.	Major

# 9 Index

Article10/04/2021

A

Abstract data model

client

server

[Adding a scoped share with an alias to a server example](#)

[ADT\\_SECURITY\\_DESCRIPTOR structure](#)

[Applicability](#)

[Applicability statement](#)

C

[Capability negotiation](#)

[Change tracking](#)

Client

[abstract data model](#)

[initialization](#)

[local events](#)

[message processing](#)

[message sequencing](#)

[sequencing rules](#)

[timer events](#)

[timers](#)

[Client side caching states](#)

[Common data types](#)

[CONNECT\\_ENUM\\_STRUCT structure](#)

[CONNECT\\_INFO\\_0\\_CONTAINER](#) structure

[CONNECT\\_INFO\\_1\\_CONTAINER](#) structure

[CONNECTION\\_INFO\\_0](#) structure

[CONNECTION\\_INFO\\_1](#) structure

Constants

[CSC states](#)

[CSC\\_CACHE\\_AUTO\\_REINT](#)

[CSC\\_CACHE\\_MANUAL\\_REINT](#)

[CSC\\_CACHE\\_NONE](#)

[CSC\\_CACHE\\_VDO](#)

D

Data model - abstract

[client](#)

[server](#)

Data model – abstract

[client](#)

[server](#)

Data types

[common - overview](#)

[Data types - common](#)

[Deleting two-phase share - example](#)

[DFS entry flags](#)

[DFS\\_Sitelist\\_Info](#) structure

[DFS\\_Sitename\\_Info](#) structure

[Disk\\_Enum.Container](#) structure

## [DISK\\_INFO structure](#)

E

Error codes ([section 2.2.2.10](#), [section 2.2.2.11](#), [section 2.2.2.12](#))

Events

[local - client](#)

[local - server](#)

[timer - client](#)

[timer - server](#)

Example of resumehandle example

Examples

[adding a scoped share with an alias to a server](#)

[example of resumehandle](#)

[overview](#)

[two-phase share deletion](#)

F

[Fields - vendor-extensible](#)

[Fields – vendor-extensible](#)

[FILE\\_ENUM\\_STRUCT structure](#)

[FILE\\_INFO\\_2 structure](#)

[FILE\\_INFO\\_2\\_CONTAINER structure](#)

[FILE\\_INFO\\_3 structure](#)

[FILE\\_INFO\\_3\\_CONTAINER structure](#)

Flags

[DFS entry](#)

[session user](#)

[software type](#)

[Full IDL](#)

[G](#)

[Glossary](#)

[I](#)

[IDL](#)

[Implementer - security considerations](#)

[Implementers – security considerations](#)

[Index of security parameters](#)

[Informative references](#)

[Initialization](#)

[client](#)

[server](#)

[Introduction](#)

[ITYPE\\_DEVICE\\_COM](#)

[ITYPE\\_DEVICE\\_CON](#)

[ITYPE\\_DEVICE\\_DISK](#)

[ITYPE\\_DEVICE\\_LPT](#)

[ITYPE\\_DEVICE\\_NUL](#)

[ITYPE\\_PATH\\_ABSD](#)

[ITYPE\\_PATH\\_ABSD\\_WC](#)

[ITYPE\\_PATH\\_ABSND](#)

[ITYPE\\_PATH\\_ABSND\\_WC](#)

[ITYPE\\_PATH\\_REL](#)

[ITYPE\\_PATH\\_REL\\_WC](#)

ITYPE\_PATH\_RELND

ITYPE\_PATH\_RELND\_WC

ITYPE\_PATH\_SYS\_COMM

ITYPE\_PATH\_SYS\_COMM\_M

ITYPE\_PATH\_SYS\_MSLOT

ITYPE\_PATH\_SYS\_MSLOT\_M

ITYPE\_PATH\_SYS\_PIPE

ITYPE\_PATH\_SYS\_PIPE\_M

ITYPE\_PATH\_SYS\_PRINT

ITYPE\_PATH\_SYS\_PRINT\_M

ITYPE\_PATH\_SYS\_QUEUE

ITYPE\_PATH\_SYS\_QUEUE\_M

ITYPE\_PATH\_SYS\_SEM

ITYPE\_PATH\_SYS\_SEM\_M

ITYPE\_PATH\_SYS\_SHMEM

ITYPE\_PATH\_SYS\_SHMEM\_M

ITYPE\_UNC

ITYPE\_UNC\_COMPNAME

ITYPE\_UNC\_SYS\_MSLOT

ITYPE\_UNC\_SYS\_PIPE

ITYPE\_UNC\_SYS\_QUEUE

ITYPE\_UNC\_SYS\_SEM

ITYPE\_UNC\_SYS\_SHMEM

ITYPE\_UNC\_WC

ITYPE\_UNC\_WC\_PATH

L

Local events

client

server

local application

disables advertising service

enables advertising service

server

deregisters

open

session

Treeconnect

looks up

null session pipes

shares

normalizes ServerName

notifies

completion of initialization

current uses of share

queries existing services

registers new

open

session

Treeconnect

service terminates

updates connection count on transport

user pauses or resumes CIFS server

LPCONNECT\_ENUM\_STRUCT

LPCONNECT\_INFO\_0\_CONTAINER

LPCONNECT\_INFO\_1\_CONTAINER

LP CONNECTION\_INFO\_0

LP CONNECTION\_INFO\_1

LPDFS\_SITELIST\_INFO

LPDFS\_SITENAME\_INFO

LPDISK\_INFO

LPFILE\_ENUM\_STRUCT

LPFILE\_INFO\_2

LPFILE\_INFO\_2\_CONTAINER

LPFILE\_INFO\_3

LPFILE\_INFO\_3\_CONTAINER

LPNET\_DFS\_ENTRY\_ID

LPNET\_DFS\_ENTRY\_ID\_CONTAINER

LPSERVER\_ALIAS\_ENUM\_STRUCT

LPSERVER\_ALIAS\_INFO\_0

LPSERVER\_INFO\_100

LPSERVER\_INFO\_1005

LPSERVER\_INFO\_101

LPSERVER\_INFO\_1010

LPSERVER\_INFO\_1016

LPSERVER\_INFO\_1017

LPSERVER\_INFO\_1018

LPSERVER\_INFO\_102

LPSERVER\_INFO\_103

LPSERVER\_INFO\_1107

LPSERVER\_INFO\_1501

LPSERVER\_INFO\_1502

LPSERVER\_INFO\_1503

LPSERVER\_INFO\_1506

LPSERVER\_INFO\_1510

LPSERVER\_INFO\_1511

LPSERVER\_INFO\_1512

LPSERVER\_INFO\_1513

LPSERVER\_INFO\_1514

LPSERVER\_INFO\_1515

LPSERVER\_INFO\_1516

LPSERVER\_INFO\_1518

LPSERVER\_INFO\_1523

LPSERVER\_INFO\_1528

LPSERVER\_INFO\_1529

LPSERVER\_INFO\_1530

LPSERVER\_INFO\_1533

LPSERVER\_INFO\_1534

LPSERVER\_INFO\_1535

LPSERVER\_INFO\_1536

LPSERVER\_INFO\_1538

LPSERVER\_INFO\_1539

LPSERVER\_INFO\_1540

LPSERVER\_INFO\_1541

LPSERVER\_INFO\_1542

LPSERVER\_INFO\_1543

LPSERVER\_INFO\_1544

LPSERVER\_INFO\_1545

LPSERVER\_INFO\_1546

LPSERVER\_INFO\_1547

LPSERVER\_INFO\_1548

LPSERVER\_INFO\_1549

LPSERVER\_INFO\_1550

LPSERVER\_INFO\_1552

LPSERVER\_INFO\_1553

LPSERVER\_INFO\_1554

LPSERVER\_INFO\_1555

LPSERVER\_INFO\_1556

LPSERVER\_INFO\_502

LPSERVER\_INFO\_503

LPSERVER\_INFO\_599

LPSERVER\_TRANSPORT\_INFO\_0

LPSERVER\_TRANSPORT\_INFO\_1

LPSERVER\_TRANSPORT\_INFO\_2

LPSERVER\_TRANSPORT\_INFO\_3

LPSERVER\_XPORT\_ENUM\_STRUCT

LPSESSION\_ENUM\_STRUCT

LPSESSION\_INFO\_0

LPSESSION\_INFO\_0\_CONTAINER

LPSESSION\_INFO\_1

LPSESSION\_INFO\_1\_CONTAINER

LPSESSION\_INFO\_10

LPSESSION\_INFO\_10\_CONTAINER

LPSESSION\_INFO\_2

LPSESSION\_INFO\_2\_CONTAINER

LPSESSION\_INFO\_502

LPSESSION\_INFO\_502\_CONTAINER

LPSHARE\_ENUM\_STRUCT

LPSHARE\_INFO\_0

LPSHARE\_INFO\_1

LPSHARE\_INFO\_1004

LPSHARE\_INFO\_1005

LPSHARE\_INFO\_1006

LPSHARE\_INFO\_1501\_I

LPSHARE\_INFO\_2

LPSHARE\_INFO\_2\_CONTAINER

LPSHARE\_INFO\_501

LPSHARE\_INFO\_501\_CONTAINER

LPSHARE\_INFO\_502\_CONTAINER

LPSHARE\_INFO\_502\_I

LPSHARE\_INFO\_503\_CONTAINER

LPSHARE\_INFO\_503\_I

LPSTAT\_SERVER\_0

LPTIME\_OF\_DAY\_INFO

M

MAX\_PREFERRED\_LENGTH

Message processing

client

server

Message sequencing

client

server

Messages

common data types

transport

Methods

NetprNameCanonicalize (Opnum 34)

NetprNameCompare (Opnum 35)

NetprNameValidate (Opnum 33)

NetprPathCanonicalize (Opnum 31)

NetprPathCompare (Opnum 32)

NetprPathType (Opnum 30)

NetrConnectionEnum (Opnum 8)

NetrDfsCreateExitPoint (Opnum 48)

NetrDfsCreateLocalPartition (Opnum 44)

NetrDfsDeleteExitPoint (Opnum 49)

NetrDfsDeleteLocalPartition (Opnum 45)

NetrDfsFixLocalVolume (Opnum 51)

NetrDfsGetVersion (Opnum 43)

NetrDfsManagerReportSiteInfo (Opnum 52)

NetrDfsModifyPrefix (Opnum 50)

NetrDfsSetLocalVolumeState (Opnum 46)

NetrFileClose (Opnum 11)

NetrFileEnum (Opnum 9)

NetrFileGetInfo (Opnum 10)

NetrpGetFileSecurity (Opnum 39)

NetrpSetFileSecurity (Opnum 40)

NetrRemoteTOD (Opnum 28)

NetrServerAliasAdd (Opnum 54)

NetrServerAliasDel (Opnum 56)

NetrServerAliasEnum (Opnum 55)

NetrServerDiskEnum (Opnum 23)

NetrServerGetInfo (Opnum 21)

NetrServerSetInfo (Opnum 22)

NetrServerStatisticsGet (Opnum 24)

NetrServerTransportAdd (Opnum 25)

NetrServerTransportAddEx (Opnum 41)

NetrServerTransportDel (Opnum 27)

NetrServerTransportDelEx (Opnum 53)

NetrServerTransportEnum (Opnum 26)

NetrSessionDel (Opnum 13)

[NetrSessionEnum](#) (Opnum 12)

[NetrShareAdd](#) (Opnum 14)

[NetrShareCheck](#) (Opnum 20)

[NetrShareDel](#) (Opnum 18)

[NetrShareDelCommit](#) (Opnum 38)

[NetrShareDelEx](#) (Opnum 57)

[NetrShareDelStart](#) (Opnum 37)

[NetrShareDelSticky](#) (Opnum 19)

[NetrShareEnum](#) (Opnum 15)

[NetrShareEnumSticky](#) (Opnum 36)

[NetrShareGetInfo](#) (Opnum 16)

[NetrShareSetInfo](#) (Opnum 17)

N

Name types

[NAMETYPE\\_COMPUTER](#)

[NAMETYPE\\_DOMAIN](#)

[NAMETYPE\\_EVENT](#)

[NAMETYPE\\_GROUP](#)

[NAMETYPE\\_MESSAGE](#)

[NAMETYPE\\_MESSAGEDEST](#)

[NAMETYPE\\_NET](#)

[NAMETYPE\\_PASSWORD](#)

[NAMETYPE\\_SERVICE](#)

[NAMETYPE\\_SHARE](#)

[NAMETYPE\\_SHAREPASSWORD](#)

NAMETYPE\_USER

NAMETYPE\_WORKGROUP

NET\_DFS\_ENTRY\_ID structure

NET\_DFS\_ENTRY\_ID\_CONTAINER structure

NetprNameCanonicalize (Opnum 34) method

NetprNameCanonicalize method

NetprNameCompare (Opnum 35) method

NetprNameCompare method

NetprNameValidate (Opnum 33) method

NetprNameValidate method

NetprPathCanonicalize (Opnum 31) method

NetprPathCanonicalize method

NetprPathCompare (Opnum 32) method

NetprPathCompare method

NetprPathType (Opnum 30) method

NetprPathType method

NetrConnectionEnum (Opnum 8) method

NetrConnectionEnum method

NetrDfsCreateExitPoint (Opnum 48) method

NetrDfsCreateExitPoint method

NetrDfsCreateLocalPartition (Opnum 44) method

NetrDfsCreateLocalPartition method

NetrDfsDeleteExitPoint (Opnum 49) method

NetrDfsDeleteExitPoint method

NetrDfsDeleteLocalPartition (Opnum 45) method

[NetrDfsDeleteLocalPartition](#) method

[NetrDfsFixLocalVolume](#) (Opnum 51) method

[NetrDfsFixLocalVolume](#) method

[NetrDfsGetVersion](#) (Opnum 43) method

[NetrDfsGetVersion](#) method

[NetrDfsManagerReportSiteInfo](#) (Opnum 52) method

[NetrDfsManagerReportSiteInfo](#) method

[NetrDfsModifyPrefix](#) (Opnum 50) method

[NetrDfsModifyPrefix](#) method

[NetrDfsSetLocalVolumeState](#) (Opnum 46) method

[NetrDfsSetLocalVolumeState](#) method

[NetrFileClose](#) (Opnum 11) method

[NetrFileClose](#) method

[NetrFileEnum](#) (Opnum 9) method

[NetrFileEnum](#) method

[NetrFileGetInfo](#) (Opnum 10) method

[NetrFileGetInfo](#) method

[NetrpGetFileSecurity](#) (Opnum 39) method

[NetrpGetFileSecurity](#) method

[NetrpSetFileSecurity](#) (Opnum 40) method

[NetrpSetFileSecurity](#) method

[NetrRemoteTOD](#) (Opnum 28) method

[NetrRemoteTOD](#) method

[NetrServerAliasAdd](#) (Opnum 54) method

[NetrServerAliasAdd](#) method

[NetrServerAliasDel \(Opnum 56\) method](#)

[NetrServerAliasDel method](#)

[NetrServerAliasEnum \(Opnum 55\) method](#)

[NetrServerAliasEnum method](#)

[NetrServerDiskEnum \(Opnum 23\) method](#)

[NetrServerDiskEnum method](#)

[NetrServerGetInfo \(Opnum 21\) method](#)

[NetrServerGetInfo method](#)

[NetrServerSetInfo \(Opnum 22\) method](#)

[NetrServerSetInfo method](#)

[NetrServerStatisticsGet \(Opnum 24\) method](#)

[NetrServerStatisticsGet method](#)

[NetrServerTransportAdd \(Opnum 25\) method](#)

[NetrServerTransportAdd method](#)

[NetrServerTransportAddEx \(Opnum 41\) method](#)

[NetrServerTransportAddEx method](#)

[NetrServerTransportDel \(Opnum 27\) method](#)

[NetrServerTransportDel method](#)

[NetrServerTransportDelEx \(Opnum 53\) method](#)

[NetrServerTransportDelEx method](#)

[NetrServerTransportEnum \(Opnum 26\) method](#)

[NetrServerTransportEnum method](#)

[NetrSessionDel \(Opnum 13\) method](#)

[NetrSessionDel method](#)

[NetrSessionEnum \(Opnum 12\) method](#)

[NetrSessionEnum method](#)

[NetrShareAdd \(Opnum 14\) method](#)

[NetrShareAdd method](#)

[NetrShareCheck \(Opnum 20\) method](#)

[NetrShareCheck method](#)

[NetrShareDel \(Opnum 18\) method](#)

[NetrShareDel method](#)

[NetrShareDelCommit \(Opnum 38\) method](#)

[NetrShareDelCommit method](#)

[NetrShareDelEx \(Opnum 57\) method](#)

[NetrShareDelEx method](#)

[NetrShareDelStart \(Opnum 37\) method](#)

[NetrShareDelStart method](#)

[NetrShareDelSticky \(Opnum 19\) method](#)

[NetrShareDelSticky method](#)

[NetrShareEnum \(Opnum 15\) method](#)

[NetrShareEnum method](#)

[NetrShareEnumSticky \(Opnum 36\) method](#)

[NetrShareEnumSticky method](#)

[NetrShareGetInfo \(Opnum 16\) method](#)

[NetrShareGetInfo method](#)

[NetrShareSetInfo \(Opnum 17\) method](#)

[NetrShareSetInfo method](#)

[Normative references](#)

O

## Overview (synopsis)

P

PADT\_SECURITY\_DESCRIPTOR

Parameters – security

Parameters - security index

Path types

PCONNECT\_ENUM\_STRUCT

PCONNECT\_INFO\_0\_CONTAINER

PCONNECT\_INFO\_1\_CONTAINER

PCONNECTION\_INFO\_0

PCONNECTION\_INFO\_1

PDFS\_Sitelist\_Info

PDFS\_Sitename\_Info

PDISK\_INFO

PFILE\_ENUM\_STRUCT

PFILE\_INFO\_2

PFILE\_INFO\_2\_CONTAINER

PFILE\_INFO\_3

PFILE\_INFO\_3\_CONTAINER

PKT\_ENTRY\_TYPE\_CAIRO

PKT\_ENTRY\_TYPE\_INSITE\_ONLY

PKT\_ENTRY\_TYPE\_LEAFONLY

PKT\_ENTRY\_TYPE\_LOCAL

PKT\_ENTRY\_TYPE\_LOCAL\_XPOINT

PKT\_ENTRY\_TYPE\_MACH\_SHARE

[PKT\\_ENTRY\\_TYPE\\_MACHINE](#)

[PKT\\_ENTRY\\_TYPE\\_NONCAIRO](#)

[PKT\\_ENTRY\\_TYPE\\_OFFLINE](#)

[PKT\\_ENTRY\\_TYPE\\_OUTSIDE\\_MY\\_DOM](#)

[PKT\\_ENTRY\\_TYPE\\_PERMANENT](#)

[PKT\\_ENTRY\\_TYPE\\_REFERRAL\\_SVC](#)

[Platform IDs](#)

[PLATFORM\\_ID\\_DOS](#)

[PLATFORM\\_ID\\_NT](#)

[PLATFORM\\_ID\\_OS2](#)

[PLATFORM\\_ID\\_OSF](#)

[PLATFORM\\_ID\\_VMS](#)

[Preconditions](#)

[Prerequisites](#)

[Product behavior](#)

[Protocol Details](#)

[overview](#)

[PSERVER\\_ALIAS\\_ENUM\\_STRUCT](#)

[PSERVER\\_ALIAS\\_INFO\\_0](#)

[PSERVER\\_INFO\\_100](#)

[PSERVER\\_INFO\\_1005](#)

[PSERVER\\_INFO\\_101](#)

[PSERVER\\_INFO\\_1010](#)

[PSERVER\\_INFO\\_1016](#)

[PSERVER\\_INFO\\_1017](#)

PSERVER\_INFO\_1018

PSERVER\_INFO\_102

PSERVER\_INFO\_103

PSERVER\_INFO\_1107

PSERVER\_INFO\_1501

PSERVER\_INFO\_1502

PSERVER\_INFO\_1503

PSERVER\_INFO\_1506

PSERVER\_INFO\_1510

PSERVER\_INFO\_1511

PSERVER\_INFO\_1512

PSERVER\_INFO\_1513

PSERVER\_INFO\_1514

PSERVER\_INFO\_1515

PSERVER\_INFO\_1516

PSERVER\_INFO\_1518

PSERVER\_INFO\_1523

PSERVER\_INFO\_1528

PSERVER\_INFO\_1529

PSERVER\_INFO\_1530

PSERVER\_INFO\_1533

PSERVER\_INFO\_1534

PSERVER\_INFO\_1535

PSERVER\_INFO\_1536

PSERVER\_INFO\_1538

PSERVER\_INFO\_1539

PSERVER\_INFO\_1540

PSERVER\_INFO\_1541

PSERVER\_INFO\_1542

PSERVER\_INFO\_1543

PSERVER\_INFO\_1544

PSERVER\_INFO\_1545

PSERVER\_INFO\_1546

PSERVER\_INFO\_1547

PSERVER\_INFO\_1548

PSERVER\_INFO\_1549

PSERVER\_INFO\_1550

PSERVER\_INFO\_1552

PSERVER\_INFO\_1553

PSERVER\_INFO\_1554

PSERVER\_INFO\_1555

PSERVER\_INFO\_1556

PSERVER\_INFO\_502

PSERVER\_INFO\_503

PSERVER\_INFO\_599

PSERVER\_TRANSPORT\_INFO\_0

PSERVER\_TRANSPORT\_INFO\_1

PSERVER\_TRANSPORT\_INFO\_2

PSERVER\_TRANSPORT\_INFO\_3

PSERVER\_XPORT\_ENUM\_STRUCT

P.getServer\_xport\_info\_0\_container

P.getServer\_xport\_info\_1\_container

P.getServer\_xport\_info\_2\_container

P.getServer\_xport\_info\_3\_container

P.session\_enum\_struct

P.session\_info\_0

P.session\_info\_0\_container

P.session\_info\_1

P.session\_info\_1\_container

P.session\_info\_10

P.session\_info\_10\_container

P.session\_info\_2

P.session\_info\_2\_container

P.session\_info\_502

P.session\_info\_502\_container

P.share\_enum\_struct

P.share\_info\_0

P.share\_info\_1

P.share\_info\_1004

P.share\_info\_1005

P.share\_info\_1006

P.share\_info\_1501\_i

P.share\_info\_2

P.share\_info\_2\_container

P.share\_info\_501

[PSHARE\\_INFO\\_501\\_CONTAINER](#)

[PSHARE\\_INFO\\_502\\_CONTAINER](#)

[PSHARE\\_INFO\\_502\\_I](#)

[PSHARE\\_INFO\\_503\\_CONTAINER](#)

[PSHARE\\_INFO\\_503\\_I](#)

[PSTAT\\_SERVER\\_0](#)

[PTIME\\_OF\\_DAY\\_INFO](#)

R

## References

informative

normative

## Relationship to other protocols

[ResumeHandle example](#)

S

## Security

[implementer considerations](#)

[parameter index](#)

## Sequencing – message

client

server

## Sequencing rules

client

server

## Server

[abstract data model](#)

initialization

local events

local application

disables advertising service

enables advertising service

server

deregisters

open

session

Treeconnect

looks up

null session pipes

shares

normalizes ServerName

notifies

completion of initialization

current uses of share

queries existing services

registers new

open

session

Treeconnect

service terminates

updates connection count on transport

user pauses or resumes CIFS server

[message processing](#)

[message sequencing](#)

[NetprNameCanonicalize \(Opnum 34\) method](#)

[NetprNameCompare \(Opnum 35\) method](#)

[NetprNameValidate \(Opnum 33\) method](#)

[NetprPathCanonicalize \(Opnum 31\) method](#)

[NetprPathCompare \(Opnum 32\) method](#)

[NetprPathType \(Opnum 30\) method](#)

[NetrConnectionEnum \(Opnum 8\) method](#)

[NetrDfsCreateExitPoint \(Opnum 48\) method](#)

[NetrDfsCreateLocalPartition \(Opnum 44\) method](#)

[NetrDfsDeleteExitPoint \(Opnum 49\) method](#)

[NetrDfsDeleteLocalPartition \(Opnum 45\) method](#)

[NetrDfsFixLocalVolume \(Opnum 51\) method](#)

[NetrDfsGetVersion \(Opnum 43\) method](#)

[NetrDfsManagerReportSiteInfo \(Opnum 52\) method](#)

[NetrDfsModifyPrefix \(Opnum 50\) method](#)

[NetrDfsSetLocalVolumeState \(Opnum 46\) method](#)

[NetrFileClose \(Opnum 11\) method](#)

[NetrFileEnum \(Opnum 9\) method](#)

[NetrFileGetInfo \(Opnum 10\) method](#)

[NetrpGetFileSecurity \(Opnum 39\) method](#)

[NetrpSetFileSecurity \(Opnum 40\) method](#)

[NetrRemoteTOD \(Opnum 28\) method](#)

[NetrServerAliasAdd \(Opnum 54\) method](#)

[NetrServerAliasDel \(Opnum 56\) method](#)

[NetrServerAliasEnum \(Opnum 55\) method](#)

[NetrServerDiskEnum \(Opnum 23\) method](#)

[NetrServerGetInfo \(Opnum 21\) method](#)

[NetrServerSetInfo \(Opnum 22\) method](#)

[NetrServerStatisticsGet \(Opnum 24\) method](#)

[NetrServerTransportAdd \(Opnum 25\) method](#)

[NetrServerTransportAddEx \(Opnum 41\) method](#)

[NetrServerTransportDel \(Opnum 27\) method](#)

[NetrServerTransportDelEx \(Opnum 53\) method](#)

[NetrServerTransportEnum \(Opnum 26\) method](#)

[NetrSessionDel \(Opnum 13\) method](#)

[NetrSessionEnum \(Opnum 12\) method](#)

[NetrShareAdd \(Opnum 14\) method](#)

[NetrShareCheck \(Opnum 20\) method](#)

[NetrShareDel \(Opnum 18\) method](#)

[NetrShareDelCommit \(Opnum 38\) method](#)

[NetrShareDelEx \(Opnum 57\) method](#)

[NetrShareDelStart \(Opnum 37\) method](#)

[NetrShareDelSticky \(Opnum 19\) method](#)

[NetrShareEnum \(Opnum 15\) method](#)

[NetrShareEnumSticky \(Opnum 36\) method](#)

[NetrShareGetInfo \(Opnum 16\) method](#)

[NetrShareSetInfo \(Opnum 17\) method](#)

[overview](#)

[sequencing rules](#)

[timer events](#)

[timers](#)

[SERVER\\_ALIAS\\_ENUM\\_STRUCT structure](#)

[SERVER\\_ALIAS\\_INFO\\_0 structure](#)

[SERVER\\_ALIAS\\_INFO\\_0\\_CONTAINER structure](#)

[SERVER\\_INFO error codes](#)

[SERVER\\_INFO\\_100 structure](#)

[SERVER\\_INFO\\_1005 structure](#)

[SERVER\\_INFO\\_101 structure](#)

[SERVER\\_INFO\\_1010 structure](#)

[SERVER\\_INFO\\_1016 structure](#)

[SERVER\\_INFO\\_1017 structure](#)

[SERVER\\_INFO\\_1018 structure](#)

[SERVER\\_INFO\\_102 structure](#)

[SERVER\\_INFO\\_103 structure](#)

[SERVER\\_INFO\\_1107 structure](#)

[SERVER\\_INFO\\_1501 structure](#)

[SERVER\\_INFO\\_1502 structure](#)

[SERVER\\_INFO\\_1503 structure](#)

[SERVER\\_INFO\\_1506 structure](#)

[SERVER\\_INFO\\_1510 structure](#)

[SERVER\\_INFO\\_1511 structure](#)

[SERVER\\_INFO\\_1512 structure](#)

[SERVER\\_INFO\\_1513 structure](#)

[SERVER\\_INFO\\_1514 structure](#)

[SERVER\\_INFO\\_1515 structure](#)

[SERVER\\_INFO\\_1516 structure](#)

[SERVER\\_INFO\\_1518 structure](#)

[SERVER\\_INFO\\_1523 structure](#)

[SERVER\\_INFO\\_1528 structure](#)

[SERVER\\_INFO\\_1529 structure](#)

[SERVER\\_INFO\\_1530 structure](#)

[SERVER\\_INFO\\_1533 structure](#)

[SERVER\\_INFO\\_1534 structure](#)

[SERVER\\_INFO\\_1535 structure](#)

[SERVER\\_INFO\\_1536 structure](#)

[SERVER\\_INFO\\_1538 structure](#)

[SERVER\\_INFO\\_1539 structure](#)

[SERVER\\_INFO\\_1540 structure](#)

[SERVER\\_INFO\\_1541 structure](#)

[SERVER\\_INFO\\_1542 structure](#)

[SERVER\\_INFO\\_1543 structure](#)

[SERVER\\_INFO\\_1544 structure](#)

[SERVER\\_INFO\\_1545 structure](#)

[SERVER\\_INFO\\_1546 structure](#)

[SERVER\\_INFO\\_1547 structure](#)

[SERVER\\_INFO\\_1548 structure](#)

[SERVER\\_INFO\\_1549 structure](#)

[SERVER\\_INFO\\_1550 structure](#)

[SERVER\\_INFO\\_1552 structure](#)

[SERVER\\_INFO\\_1553 structure](#)

[SERVER\\_INFO\\_1554 structure](#)

[SERVER\\_INFO\\_1555 structure](#)

[SERVER\\_INFO\\_1556 structure](#)

[SERVER\\_INFO\\_502 structure](#)

[SERVER\\_INFO\\_503 structure](#)

[SERVER\\_INFO\\_599 structure](#)

[SERVER\\_TRANSPORT\\_INFO\\_0 structure](#)

[SERVER\\_TRANSPORT\\_INFO\\_1 structure](#)

[SERVER\\_TRANSPORT\\_INFO\\_2 structure](#)

[SERVER\\_TRANSPORT\\_INFO\\_3 structure](#)

[SERVER\\_XPORT\\_ENUM\\_STRUCT structure](#)

[SERVER\\_XPORT\\_INFO\\_0\\_CONTAINER structure](#)

[SERVER\\_XPORT\\_INFO\\_1\\_CONTAINER structure](#)

[SERVER\\_XPORT\\_INFO\\_2\\_CONTAINER structure](#)

[SERVER\\_XPORT\\_INFO\\_3\\_CONTAINER structure](#)

[SESS\\_GUEST](#)

[SESS\\_NOENCRYPTION](#)

Session user flags

[SESSION\\_ENUM\\_STRUCT structure](#)

[SESSION\\_INFO\\_0 structure](#)

[SESSION\\_INFO\\_0\\_CONTAINER structure](#)

[SESSION\\_INFO\\_1 structure](#)

[SESSION\\_INFO\\_1\\_CONTAINER structure](#)

[SESSION\\_INFO\\_10 structure](#)

[SESSION\\_INFO\\_10\\_CONTAINER structure](#)

[SESSION\\_INFO\\_2 structure](#)

[SESSION\\_INFO\\_2\\_CONTAINER structure](#)

[SESSION\\_INFO\\_502 structure](#)

[SESSION\\_INFO\\_502\\_CONTAINER structure](#)

[Sessionclient](#)

[Share types](#)

[SHARE\\_ENUM\\_STRUCT structure](#)

[SHARE\\_INFO error codes](#)

[SHARE\\_INFO\\_0 structure](#)

[SHARE\\_INFO\\_0\\_CONTAINER structure](#)

[SHARE\\_INFO\\_1 structure](#)

[SHARE\\_INFO\\_1\\_CONTAINER structure](#)

[SHARE\\_INFO\\_1004 structure](#)

[SHARE\\_INFO\\_1005 structure](#)

[SHARE\\_INFO\\_1006 structure](#)

[SHARE\\_INFO\\_1501\\_I structure](#)

[SHARE\\_INFO\\_2 structure](#)

[SHARE\\_INFO\\_2\\_CONTAINER structure](#)

[SHARE\\_INFO\\_501 structure](#)

[SHARE\\_INFO\\_501\\_CONTAINER structure](#)

[SHARE\\_INFO\\_502\\_CONTAINER structure](#)

[SHARE\\_INFO\\_502\\_I structure](#)

[SHARE\\_INFO\\_503\\_CONTAINER structure](#)

[SHARE\\_INFO\\_503\\_I structure](#)

[Software type flags](#)

[Standards assignments](#)

[STAT\\_SERVER\\_0 structure](#)

[Structures](#)

[STYPE\\_CLUSTER\\_DFS](#)

[SType\\_CLUSTER\\_FS](#)

[SType\\_CLUSTER\\_SOFS](#)

[SType\\_DEVICE](#)

[SType\\_DISKTREE](#)

[SType\\_IPC](#)

[SType\\_PRINTQ](#)

[SType\\_SPECIAL](#)

[SType\\_TEMPORARY](#)

[SV\\_TYPE\\_AFP](#)

[SV\\_TYPE\\_ALL](#)

[SV\\_TYPE\\_ALTERNATE\\_XPORT](#)

[SV\\_TYPE\\_BACKUP\\_BROWSER](#)

[SV\\_TYPE\\_CLUSTER\\_NT](#)

[SV\\_TYPE\\_CLUSTER\\_VS\\_NT](#)

[SV\\_TYPE\\_DCE](#)

[SV\\_TYPE\\_DFS](#)

[SV\\_TYPE\\_DIALIN\\_SERVER](#)

[SV\\_TYPE\\_DOMAIN\\_BAKCTRL](#)

[SV\\_TYPE\\_DOMAIN\\_CTRL](#)

[SV\\_TYPE\\_DOMAIN\\_ENUM](#)

[SV\\_TYPE\\_DOMAIN\\_MASTER](#)

[SV\\_TYPE\\_DOMAIN\\_MEMBER](#)

[SV\\_TYPE\\_LOCAL\\_LIST\\_ONLY](#)

[SV\\_TYPE\\_MASTER\\_BROWSER](#)

[SV\\_TYPE\\_NOVELL](#)

[SV\\_TYPE\\_NT](#)

[SV\\_TYPE\\_POTENTIAL\\_BROWSER](#)

[SV\\_TYPE\\_PRINTQ\\_SERVER](#)

[SV\\_TYPE\\_SERVER](#)

[SV\\_TYPE\\_SERVER\\_MFPN](#)

[SV\\_TYPE\\_SERVER\\_NT](#)

[SV\\_TYPE\\_SQLSERVER](#)

[SV\\_TYPE\\_TERMINALSERVER](#)

[SV\\_TYPE\\_TIME\\_SOURCE](#)

[SV\\_TYPE\\_WFW](#)

[SV\\_TYPE\\_WINDOWS](#)

[SV\\_TYPE\\_WORKSTATION](#)

[SV\\_TYPE\\_XENIX\\_SERVER](#)

T

[TIME\\_OF\\_DAY\\_INFO structure](#)

Timer events

client

server

Timers

[client](#)

[server](#)

[Tracking changes](#)

[Transport](#)

[Transport – message](#)

[Two-phase share deletion example](#)

[U](#)

[Unions](#)

[V](#)

[Vendor-extensible fields](#)

[Versioning](#)

[W](#)

[Windows error codes](#)