[MS-WSO]:
Windows Protocols System Overview

Abstract

The Windows Protocols system includes the set of protocols that are part of the Microsoft Communications Protocol Program (MCPP). This includes protocols implemented in a Microsoft Windows 2000 Professional, Windows XP, Windows Vista, or successor desktop operating system that are used to interoperate or communicate with a Microsoft Server operating system product. This document provides a system overview for protocols and systems described in the protocols technical documents (TDs), Protocol Family system documents (PFSDs), and Defined Tasks system documents (DTSDs) that are included in the MCPP program. It provides a system overview of how these systems and protocols relate to each other.

Date	Comments
2009.06.30	First draft.

Table of Contents

1 Introduction

Microsoft works with many other companies and industry initiatives to enable Microsoft products to interoperate with other networks and network services. Network protocols are developed and promoted by a variety of formal standards bodies, industry consortia, and individual companies worldwide. Microsoft actively participates and contributes to the standardization process in many standards bodies and develops implementations to make the Windows operating system interoperable with other products that implement these protocols.

Microsoft makes available specific client-server protocols implemented in Windows, together with whatever intellectual property rights it may have in those protocols, consistent with its obligations under the antitrust case settlement. Some of those protocols are documented and published on Microsoft websites such as MSDN, or through standards organizations or other third-party sources. Communications protocols are available for licensing through the Microsoft Communications Protocol Program (MCPP) and other programs, and through a royalty-free implementation license.

This document provides a system overview for protocols and systems described in the protocols technical documents (TDs), Protocol Family system documents (PFSDs), and Defined Tasks system documents (DTSDs) that are included in the MCPP program. It provides a system overview of how these systems and protocols relate to each other.

This document is organized as follows:

- Section 1, "Introduction", describes what is covered in this document, provides a list of terms defined in this document and pointers to terms used in this document but defined elsewhere in the documentation set, and provides a list of references that apply to the overall Windows Protocols System.
- Section 2, "Overview", introduces the Windows Protocols System. It provides a high-level map of how the systems and individual protocols in MCPP relate to each other.
- Section 3, "Foundation", presents the background information that the reader would reasonably need to know to understand and implement the diverse set of protocols that exist in the MCPP document set. It includes concepts related to each of the categories of protocols, purposes of the Windows Protocols System, and a list of the use cases described in each System Document.
- Section 4, "System Context", describes the Windows environment for the MCPP protocols and black box diagrams that depict the relationships of the protocol family systems with external components and the Windows environment.
- Section 5, "System Architecture", describes the white box relationships, including the protocol layering of the MCPP protocols at the system level and the interrelationships among its components, consumers, and dependencies. It lists the abstract data model elements that are shared at the system level in each of the protocol family and tasks systems.
- Section 6, "System Details", lists the examples covered in the Architectural Details sections in the System Documents.
- Appendix B, "Scenarios", provides examples for the following subjects:
  - Power Shell
  - Branch Caching

1.1 Glossary

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

Active Directory (AD)

application

certificate

certification authority (CA)

certificate revocation list (CRL)

certificate template

client

Distributed File System (DFS) client

directory

domain

domain controller (DC)

enrollment

Group Policy

health policy server

Lightweight Directory Access Protocol (LDAP)

Network Access Protection (NAP)

Network Access Protection (NAP) client

metafile

named pipe

player

print client

printer driver

print job

print queue

print server

protocol data unit (PDU)

reference identifier (RID)

Remote Desktop Protocol (RDP)

remote procedure call (RPC)

reference identifier (RID)

schema object

security account manager (SAM)

security identifier (SID)

server

Server Message Block (SMB)

shell

smart card

terminal server

Terminal Services

TLS

transaction

transaction manager

trust

tunnel

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

Collaboration Client

ILS Server

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

game session

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

Participant

Resource

resource manager

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

Caller ID

Transmitting Station ID (TSID) [aka Transmission ID]

Fax Service Provider (FSP)

Fax Job

Fax Queue

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

DFS Service

File Client

file server

NFS File Client

NFS File Service

SMB File Client

SMB File Service

SMB Access Protocols

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

DirectPlay System

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

message

outgoing queue

queue

queue manager

system queue

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

asynchronous messaging

network address

The following terms are defined in defined in [MS-PCCRC]

Content

Content Server

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

Hosted cache

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

Print Spooler

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

Multipoint Communication Service (MCS)

The following terms are defined in [MS-RDPCR2]:

desktop

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

dynamic virtual channel (DVC)

RDP Client

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

static virtual channel

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

license

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

Print Processor

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

object store

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

system health validator (SHV)

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

Internet Locator Service (ILS)

The following terms are defined in [MS-TPSO]

two-phase commit protocol

transaction trees

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

clipboard redirection

firewall

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

downstream server (DSS)

update

upstream server (USS)

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

client computer

metadata

target group

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

certificate enrollment policy

The following terms are defined in [RFC2753]:

policy enforcement point (PEP)

policy decision point (PDP)

The following terms are newly defined in this document:

BranchCache™: A Content Caching and Retrieval feature, enables content from file and web servers on a wide area network (WAN) to be cached on computers at a local branch office. Available in 2 modes: Hosted Cache and Distributed Cache.

Hosted Cache: A centralized cache composed of blocks of data added by peers

Distributed Cache: A cache composed of blocks of data hosted on multiple peers acting in cooperation

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

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

1.2 References

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

1.2.1 Normative References

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

[FIPS180-2] Federal Information Processing Standards Publication, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[FIPS197] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES)", November 2001, http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[MC-DPLNAT] Microsoft Corporation, "DirectPlay 8 Protocol: NAT Locator Specification" September 2007.

[MS-ADSO] Microsoft Corporation, "Active Directory System Overview", to be published 2009.

[MS-AUTHSO] Microsoft Corporation, "Authentication Services System Overview", to be published 2009.

[MS-CAESO] Microsoft Corporation, "Certificate Autoenrollment System Overview", to be published 2009.

[MS-CASO] Microsoft Corporation, "Certification Authority System Overview", May 2009.

[MS-CIFS] Microsoft Corporation, "Common Internet File System Specification", to be published 2009.

[MS-COMA] Microsoft Corporation, "Component Object Model Plus (COM+) Remote Administration Protocol Specification", July 2007.

[MS-CSSO] Microsoft Corporation, "Collaboration Services System Overview", to be published 2009.

[MS-DCOM] Microsoft Corporation, "Distributed Component Object Model (DCOM) Remote Protocol Specification", March 2007.

[MS-DISO] Microsoft Corporation, "Domain Interactions System Overview", to be published 2009.

[MSDN-NCALRPC] Microsoft Corporation, "ncalrpc Attribute", http://msdn.microsoft.com/en-us/library/aa367115.aspx

[MS-DTYP] Microsoft Corporation, "Windows Data Types", February 2008.

[MS-EMFPLUS] Microsoft Corporation, "Enhanced Metafile Format Plus Extensions Specification", June 2007.

[MS-EVEN] Microsoft Corporation, "EventLog Remoting Protocol Specification" January 2007.

[MS-EVEN6] Microsoft Corporation, "EventLog Remoting Protocol Version 6.0 Specification", January 2007.

[MS-FAX] Microsoft Corporation, "Fax Server and Client Remote Protocol Specification", March 2007.

[FIPS140] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules", December 2002, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

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

[MS-FSSO] Microsoft Corporation, "File Access Services System Overview", to be published 2009.

[MS-GPSO] Microsoft Corporation, "Group Policy System Overview", February 2009.

[MS-LWSSP] Microsoft Corporation, "Lightweight Web Services Security Profile", December 2008.

[MS-MGSO] Microsoft Corporation, "DirectPlay System Overview", February 2009.

[MS-MQSO] Microsoft Corporation, "Message Queuing System Overview", May 2009.

[MS-MSSO] Microsoft Corporation, "Media Streaming Server System Overview", to be published 2009.

[MS-NAPSO] Microsoft Corporation, "Network Access Protection System Overview", to be published 2009.

[MS-PCCRC] Microsoft Corporation, "Peer Content Caching and Retrieval: Content Identification", June 2009.

[MS-PCCRD] Microsoft Corporation, "Peer Content Caching and Retrieval Discovery Protocol Specification", June 2009.

[MS-PCCRR] Microsoft Corporation, "Peer Content Caching and Retrieval: Retrieval Protocol Specification", June 2009.

[MS-PCCRTP] Microsoft Corporation, "Peer Content Caching and Retrieval: Hypertext Transfer Protocol (HTTP) Client Extensions", June 2009.

[MS-PCHC] Microsoft Corporation, "Peer Content Caching and Retrieval: Hosted Cache Protocol Specification", June 2009.

[MS-PSSO] Microsoft Corporation, "Print Services System Overview", to be published 2009.

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

[MS-RMPR] Microsoft Corporation, "Rights Management Services (RMS): Client-to-Server Protocol Specification", March 2007.

[MS-RMSO] Microsoft Corporation, "Rights Management Services System Overview", May 2009.

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", June 2006.

[MS-RPCH] Microsoft Corporation, "Remote Procedure Call over HTTP Protocol Specification", January 2007.

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Version 2 Protocol Specification", June 2009.

[MS-SOH] Microsoft Corporation, "Statement of Health for Network Access Protection (NAP) Protocol Specification", January 2007.

[MS-TPSO] Microsoft Corporation, "Transaction Processing Services System Overview", February 2009.

[MS-TSSO] Microsoft Corporation, "Terminal Services System Overview", May 2009.

[MS-WMSO] Microsoft Corporation, "Windows Management System Overview", April 2009.

[MS-WSMAN] Microsoft Corporation, "Web Services Management Protocol Extensions for Windows Server 2003", March 2007.

[MS-WSUSO] Microsoft Corporation, "Windows Server Update Services System Overview", May 2009.

[MS-WUSP] Microsoft Corporation, "Windows Update Services: Client-Server Protocol Specification", September 2007.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982, http://www.ietf.org/rfc/rfc0821.txt.

[RFC822] Crocker, D.H., "Standard for ARPA Internet Text Messages", STD 11, RFC 822, August 1982, http://www.ietf.org/rfc/rfc0822.txt.

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

[RFC2246] Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt.

[RFC3302] Parsons, G., and Rafferty, J., "Tag Image File Format (TIFF) - image/tiff MIME Sub-Type Registration", RFC 3302, September 2002, http://www.ietf.org/rfc/rfc3302.txt.

[RFC4086] Eastlake III, D., Schiller, J., and Crokcer, S., "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005, http://www.ietf.org/rfc/rfc4086.txt.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.txt.

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, http://www.ietf.org/rfc/rfc4122.txt.

[WS-Discovery] Beatty, J., Kakivaya, G., Kemp D., et al., "Web Services Dynamic Discovery (WS-Discovery)", April 2005, http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf.

1.2.2 Informative References

[BROWN-WINSEC] Brown, Keith, "Programming Windows Security", Addison-Wesley Professional, 2000, ISBN 0201604426.

[DAVIES-TCPIP] Joseph Davies, "Windows Server 2008 TCP/IP Protocols and Services", published January, 2008, ISBN 9780735624474.

[DIALOGUE] Bryant, B. and Ts'o, T., "Designing an Authentication System: A Dialogue in Four Scenes", February 1997, http://web.mit.edu/kerberos/www/dialogue.html.

[GRAY] Gray, J. and Reuter, A., "Transaction Processing: Concepts and Techniques", San Mateo, CA: Morgan Kaufmann Publishers, 1993, ISBN: 1558601902.

[HOWARD-SECODE] Howard, Michael and LeBlanc, David, "Writing Secure Code", Microsoft Press, 2002, ISBN 0735617228.

[KERBPAC] Brezak, J., "Utilizing the Windows 2000 Authorization Data in Kerberos Tickets for Access Control to Resources", February 2002, http://msdn.microsoft.com/en-us/library/aa302203.aspx.

[MSDFS] Microsoft Corporation, "How DFS Works", March 2003, http://technet2.microsoft.com/WindowsServer/en/library/a9096e88-1634-4da6-b820-537341d349061033.mspx.

[MSFT-LEGALMCPP] Microsoft Corporation, "Microsoft Communications Protocol Program", http://www.microsoft.com/about/legal/intellectualproperty/protocols/mcpp.mspx.

[RUSSINOVICH-WININTERNALS] Russinovich, M. and Solomon, D., "Microsoft Windows Internals, Fourth Edition", Microsoft Press, 2005, ISBN: 0735619174.

[WSTrust] IBM, Microsoft, Nortel, VeriSign, "WS-Trust V1.0", February 2005, http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf

2 Overview

This section provides an overview of the protocol families, defined tasks, and individual protocols that make up the Windows Protocols System.

2.1 System Summary

Protocols of the Windows Protocols System are part of the Microsoft Communications Protocol Program (MCPP), which includes protocols implemented in a Microsoft Windows 2000 Professional, Windows XP, Windows Vista, or Windows 7 desktop operating system that are used to operate or communicate with a Microsoft server-side operating system product.

Microsoft client-server communications uses both industry-standard protocols and protocols licensed under the MCPP program. Protocols included in the MCPP program are either an extension of an industry standard protocol (such as the Kerberos Interactive Logon Protocol Extensions) or a Microsoft proprietary protocol. Most MCPP protocols are extensions to industry-standard protocols.

In cases where the MCPP protocol is an extension of an industry standard protocol, the MCPP documentation describes the nature and intent of the extension. In general, it does not describe that part of the protocol already defined in the standards documentation. For Microsoft proprietary protocols, the MCPP documentation defines the complete protocol.

The MCPP program licenses protocols for the following MCPP licensing tasks:

- Authentication Services Server
- Certificate Services Server
- Collaboration Server
- Digital Rights Management Server
- File Server
- General Server
- Health Certificate Server
- Media Streaming Server
- Multiplayer Games Server
- Print/Fax Server
- Proxy/Firewall/NAT Server
- Rights Management Server
- Systems Management Server
- Terminal Server
- Virtual Private Network Server
- Web Server
- Windows Update Services Server

For the most up-to-date list of MCPP tasks and associated protocols, please refer to the MCPP website at http://www.microsoft.com/about/legal/intellectualproperty/protocols/mcpp.mspx. Microsoft updates this website regularly to include updated and new MCPP protocols as detailed in the MCPP license agreement.

The following section summarizes the protocol family systems and defined tasks systems that are part of the overall Windows Protocols System.

2.2 List of Windows Protocol Systems and Tasks

The **Active Directory System** provides a directory service. A directory service is a service that stores and organizes directory objects in a centralized, hierarchical data store. This hierarchical organization of objects is called the directory. A directory object is an object that contains one or more attributes. Each attribute can have one or more values. Directory objects are identified by a name that is unique among all directory objects in the directory service. The directory objects are organized in a hierarchical manner with regards to other directory objects. The Active Directory System provides the foundation for authentication services in a domain environment. Additionally, it is used as a component by other systems such as Group Policy and Print Services. Active Directory is a directory service that provides for the centralized storage of identity and account information, as well as storage for other forms of data, such as group policies and printer location information. A directory service contains one or more servers, known as directory servers, in which directory objects can be created, queried for, modified, and deleted. Each directory object is a collection of attributes, each of which contains one or more values. One common directory object is the user object, which represents an account and stores information such as the user name and password.

The **Authentication Services System** includes several authentication tasks between client and server computers and domain controllers. It covers the components necessary for authentication for several specific tasks of interactive logon, HTTP Web access, file system services, and remote operations. Authentication is the action of proving identity to a network service or a resource provider. Authentication plays a central role in the Windows operating system as a basis for proof of identity and ability to control access through authorization. The authentication process can use services provided by clients, servers, and domain controllers. The servers and domain controllers work in conjunction to provide management of accounts; the client provides services to manage the credentials to be used to prove identity to the server computer. Domain controllers provide directory services, specifically Active Directory, which is the default Windows technology for storing identity information.

The **Certificate Autoenrollment System** includes one task for automatically enrolling and renewing digital certificates automatically. System administrators usually perform this task manually. Autoenrollment automatically handles certificate enrollment and the re-enrollment of expired certificates, which relieves the administrator from performing this task manually. Microsoft networks and protocols often require the use of digital certificates for encryption and authentication. Implementing the Computer Certificate Autoenrollment Task enables a system to enroll and re-enroll certificates automatically. Autoenrollment handles all of the details for enrollment and re-enrollment of certificates. Autoenrollment serves a central role in client and server relationships that rely on certificate enrollment. Autoenrollment determines what policies are available for certificate enrollment, the set of certificates specified through these policies, and what certificates can be issued based on the templates in these policies.

The **Certification Authority System** uses public key cryptography to issue certificates that can be used for a variety of purposes, including encryption and authentication. A certification authority (CA) issues certificates and confirms to other entities that the certificate is valid. People, computers, and applications, can all be issued certificates from the CA. Certificate holders can use the private key to digitally encrypt data, to digitally sign documents, and to identify themselves. A public key infrastructure (PKI) supports public key cryptography within and between organizations. A PKI consists of digital certificates, key pairs, CAs, and other registration authorities. A certificate is a digital statement issued by a CA that vouches for the identity of the certificate holder; a certificate binds a public key and a collection of attributes to the certificate holder. The certificate can be freely shared with other entities.

The **Collaboration Services System** includes two sets of logical procedures (Defined Tasks) necessary for Collaboration Clients to locate each other on a network using the directory facilities provided by an ILS Server. Software used for collaboration includes applications such as application sharing, e-mail, whiteboarding, sharing a calendar, instant messaging, text chat, and remote desktop. **NetMeeting** is an example of the Microsoft implementation of collaboration software. One of the common Tasks used in collaboration applications includes methods that allow clients to locate each other. One of the solutions is to register their location in a central store so that other **Collaboration Clients** can find them. The **Collaboration Services System** Tasks use the Telephony API Internet Locator Service Data block protocol specified in [MS-TAIL] as the key protocol for communications with the central store.

The **Domain Interactions System** includes the Windows protocols that are used together to maintain a relationship with the domain. This includes protocols that are used to communicate with a domain controller and maintain state, protocols that are used to augment authentication and authorization actions, and protocols that are used to interact with domain controllers. Microsoft Windows networks are often configured with a domain controller providing centralized storage of accounts and administration of many machines. Many network related operations depend on domains in order to complete various tasks. The Domain Interactions System describes some of these tasks, including:

- Locating a domain controller using DNS and NetBIOS.
- Joining a domain using a predefined account.
- Joining a domain by creating an account via the Security Account Manager (SAM) RPC interface.
- Joining a domain by creating an account via the Lightweight Directory Access Protocol (LDAP) protocol.
- Removing a domain member.

The **File Access Services System** includes the protocols that enable network file access and sharing in Microsoft Windows operating environments. File Access Services allow a client computer to discover, access, and share files that are hosted on, and made available by, a server computer. Most businesses and many personal computing environments have a need for file hierarchies stored in a file system (**object store**) on one computer to be accessed and manipulated by applications on other computers. This scenario is typically referred to as file sharing or remote file access. A key goal is to ensure application compatibility by providing broadly the same semantics to clients as if the shared object store were local to them. Sharing the files through a protocol with more limited semantics, such as FTP or HTTP, would not provide this compatibility. The File Access Services System addresses these needs.

The **Group Policy System** enables a Group Policy Client to retrieve policy settings from a Group Policy Server and enables administrative tools to retrieve, create, update, and delete policy settings. Group Policy enables administrators to define and manage desired computer configurations or policy settings for a large number of users and computers within an Active Directory environment. Administrators can define policy settings once and rely on the Windows operating system to enforce that policy.

The **Message Queuing System** is a communications service that enables reliable and secure asynchronous messaging between applications over a variety of deployment topologies. The Message Queuing System temporally decouples the act of sending a message from the act of receiving that message, allowing applications to communicate even if their execution lifetimes do not overlap. Applications send messages to a queue and/or receive messages from a queue. The queue provides persistence of the messages, enabling them to survive across application restarts. This abstraction enables an application to send messages even if the receiving application is not executing or is unreachable due to a network outage.

The **Media Streaming Server (MSS) System** enables the creation, distribution, and playback of audio and video **content**. It enables administrators and content providers to create media solutions for various uses such as corporate communications, training and education, e-commerce, and commercial broadcast. The main physical components of MSS include a computer running a media encoder, a server running as a media server, and several client computers running media play clients. The encoder converts both live and prerecorded audio and video content to a media format, the server distributes the content over a network or the Internet and the media player receives the content. In order to scale and meet network demands, the system can also include cache and proxy servers, and distribution servers. For E-commerce scenarios, MSS enables scenarios ranging from live broadcast playback to on-demand playback and may require the support of Digital Rights Management components to provide the administrator with the ability to securely encrypt content that is being broadcasted and downloaded.

The **Multiplayer Games System** is designed to transport game and user data to support multiplayer gaming scenarios. The protocols in this system provide game session management as well as functionality for controlling options for sending data and voice. Control options for data include reliability, guaranteeing data delivery sequencing, and coalescence of packets. The DirectPlay System also provides functions for using network address translation (NAT).

The **Network Access Protection System** includes a variety of tasks that allow for the communication of information related to the health of a system, evaluation of health information, restricted network access, and correction of health state, including the following:

- NAP clients collect and communicate their health status information to NAP health policy servers.
- NAP health policy servers determine overall health compliance and the prescribed actions and remediation measures that a NAP client must take, if found to be noncompliant with policy, so that it can become compliant.
- NAP health policy servers collect and communicate prescribed actions and remediation measures to NAP clients.
- NAP health policy servers communicate network access restrictions to policy enforcement points for noncompliant NAP clients.
- NAP clients receive prescribed actions and remediation measures and correct their system health.

NAP is a distributed system of cooperating clients and servers, which collectively ensures that individual hosts and the corporate network as a whole are healthy. "Healthy" means that the systems have the updates and software installed and they are configured in the manner prescribed by system health policies. The goal of the NAP System is to ensure that individual hosts and the IT infrastructure as a whole are robust and resistant to attacks and malware outbreaks.

The **Print Services System** provides functionality for managing a distributed print infrastructure of Print Clients, print jobs, and shared printers. One or more printers are shared among multiple Print Clients using a Print Server. Printers are represented as Print Queues located on Print Servers. A Print Client has one or more connections to Print Queues shared by one or more Print Servers, comprising a simple hub-spoke server-client model. The Print Services System enables a large number of people to print to a small number of printers, enables an administrator to enforce which printers can be used by each user and when they can be used, and automates the process of installing and updating necessary printer drivers on numerous computers that send jobs to remote printers. The Print Services System can be deployed within a domain-based network, or in a home or workgroup environment. With Windows implementations, each Print Client can also act as a Print Server and share locally connected printers with other Print Clients.

The **Rights Management Services System** provides protection of information through persistent usage policies. This enables usage rights to be enforced after the information is accessed by an authorized recipient, such as restricting copying, printing, or forwarding. RMS enables enforcement of corporate policy governing the control and dissemination of confidential or proprietary information. Rights management allows individuals and administrators to encrypt and specify access permissions to various types of data, including documents and e-mail messages. This helps prevent sensitive information from being accessed and used by unauthorized people. After permissions to content have been restricted by using rights management, the access and usage restrictions are enforced no matter where the information is, because the permissions are stored in the content itself.

The **Transaction Processing Services System** provides transaction processing services for systems that require transactions to be coordinated in a distributed system. Transaction processing is designed to maintain a computation system in a known, consistent state. It allows multiple individual operations to be linked together as a single, indivisible operation: an atomic transaction. Broadly speaking, transaction processing involves updating data, which may be distributed across multiple systems, so that either all the changes happen or none of the changes happen. Resources may be data, such as rows in a database, or logical entities, such as the execution state of a program. Resources changed by a transaction may be in separate systems.

The **Terminal Services System** provides functionality for securely connecting remote clients and servers, for channeling communication between components of remote clients and servers, and for managing servers. The Terminal Services System implements the Remote Desktop Protocol (RDP), which is a multi-channel protocol that allows users of a remote client to connect to a server over a network. This multi-channel capability enables the use of separate channels, called virtual channels, to carry different types of data including presentation data, highly-encrypted data (such as keyboard and mouse user input), device communication, and licensing information.

The **Windows Management System (WMS)** provides an infrastructure that enables a user or application to inspect, modify, and monitor an asset's resources remotely. In WMS, resources are represented as objects, following the Common Information Model (CIM) provides multiple network protocols for accessing resources. The protocols vary in capability.

The **Windows Server Update Services System** enables software update discovery by computers, delivery of relevant updates to computers, update distribution controls for administrators, and monitoring of software update activity between an update client and update server. A software update can be an update to an application or distribution of an application or driver for a hardware device. The Windows Server Update client is capable of detecting software updates that are applicable from the available set of updates on the server, installing such updates, and reporting installation activity back to the server. This system requires communication between the Windows Server Update Services client and server to enable clients to discover software updates available on the server. In addition, it also requires communication between servers to propagate software update information, the updates, and administrative intent in a hierarchical deployment of the system.

2.3 Relevant Standards

The Windows Protocols System implements and extends many standards. Appendix D, "Standards Implemented in Windows" provides a list with descriptions of all standards used or extended in Windows.

3 Foundation

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

3.1 Background Knowledge and System-Specific Concepts

This section summarizes:

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

The Windows Protocols System documentation assumes the reader has knowledge of publicly available standard specifications, network programming art, and Windows distributed systems concepts such as:

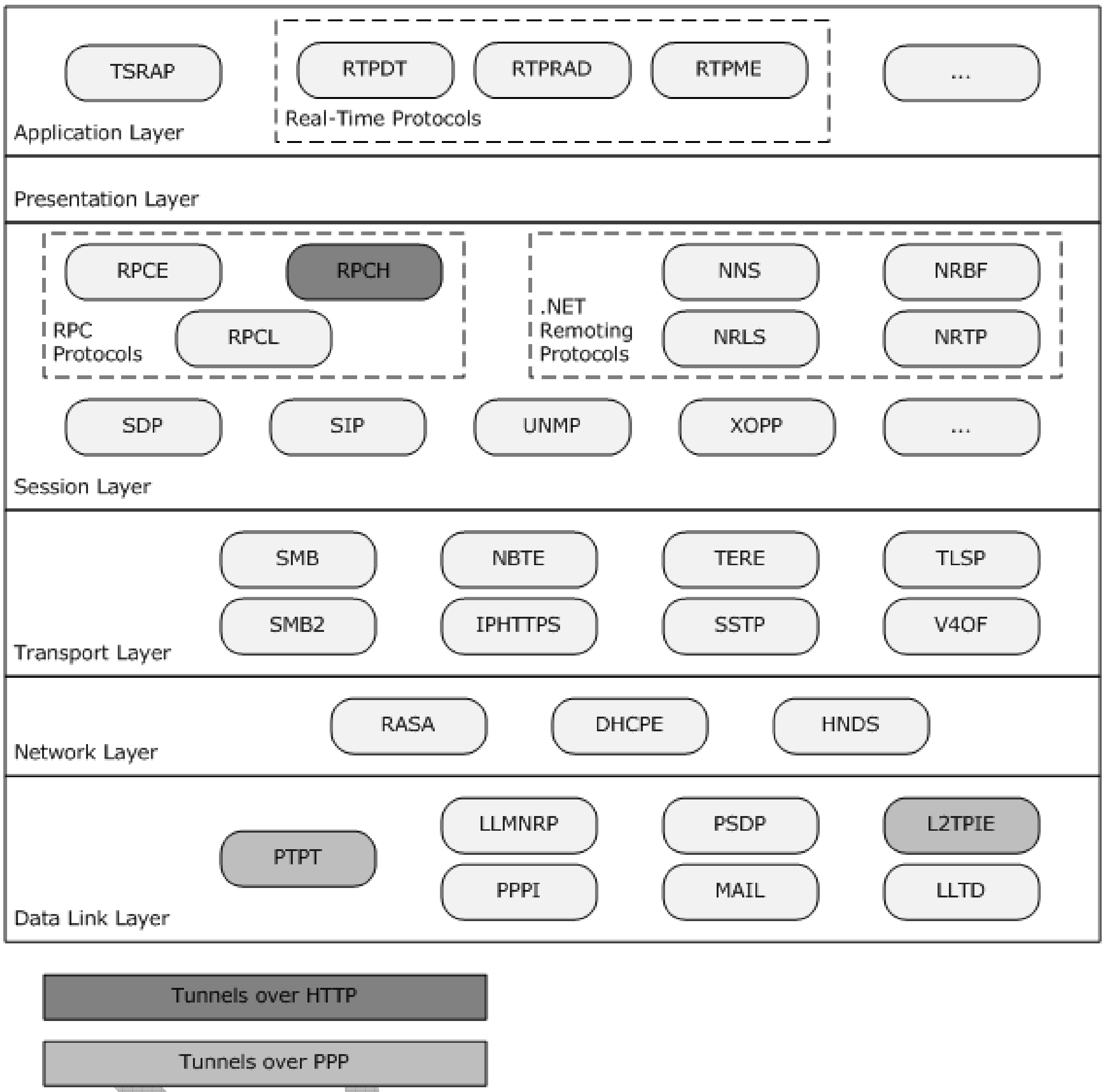
- TCP/IP
- HTTP
- SOAP
- UNC

Foundational knowledge of the following concepts, presented later in this section, is also assumed:

- Remote procedure calls (RPC)
- Named pipes
- Domain services
- Authentication and authorization

3.1.1 Networking and Transport Concepts

Microsoft provides a number of protocols at different layers of the OSI Networking Model. The following diagram shows where the Microsoft networking protocols fit into that model.



Details of the RPC Protocols are discussed in the following section.

3.1.1.1 Remote Procedure Calls

This section provides an overview of remote procedure calls and how they are implemented in Windows.

3.1.1.1.1 Remote Procedure Call Model

A remote procedure call (RPC) is a secure inter-process communication (IPC) mechanism that enables data exchange and invocation of functionality residing in a different process. That different process can be on the same machine, on the local area network, or across the Internet. This section explains the RPC programming model and the model for distributed systems that can be implemented by using RPC.

RPC supports 64-bit editions of Windows. In Windows XP, there are three types of processes: native 32-bit processes, native 64-bit processes, and 32-bit processes running under the 32-bit process emulator on a 64-bit system (often referred to as WOW6432 processes). Using RPC, developers can transparently communicate between different types of process; RPC automatically manages process differences behind the scenes.

RPC was initially developed as an extension to Open Software Foundation (OSF) RPC (later it became the distributed Computer Environment aka DCE RPC see [C706]). With the exception of some of its advanced features, The Microsoft implementation of RPC is interoperable with other vendors' implementations of OSF /DCE RPC.

This section also provides an overview of RPC components and their operation. The information is presented in the following topics:

- Programming Model
- Distributed Systems Model
- How RPC Works
- Microsoft RPC Components
- Microsoft RPC Security Model

3.1.1.1.2 Programming Model

In the early days of computer programming, each program was written as a large monolithic chunk that was filled with goto statements. Each program had to manage its own input and output to different hardware devices. As the programming discipline matured, this monolithic code was organized into procedures and the most commonly used procedures were packed in libraries for sharing and reuse.

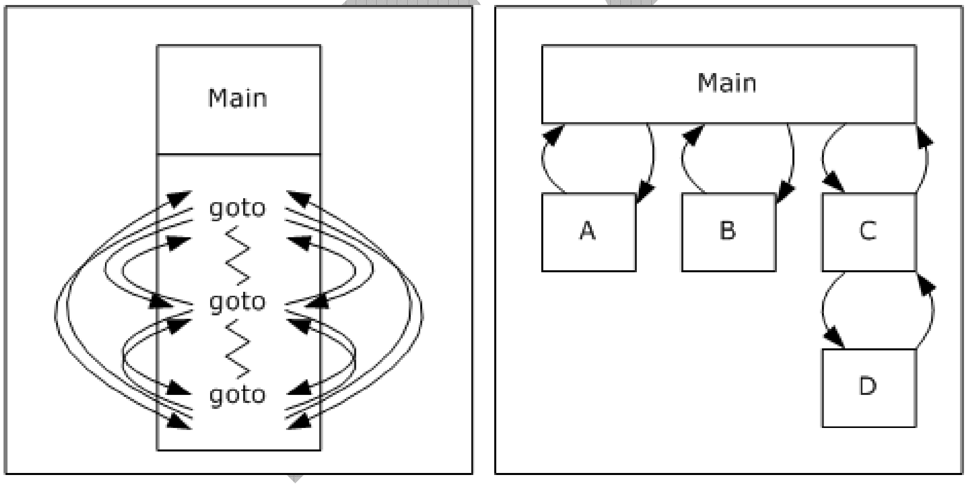


Figure 1: Monolithic vs. procedural implementation model

In procedure-oriented programming languages such as C, the main procedure relates to all other procedures as black boxes. For example, the main procedure cannot find out how procedures A, B, and X do their work. The main procedure only calls another procedure; it has no information about how that procedure is implemented.

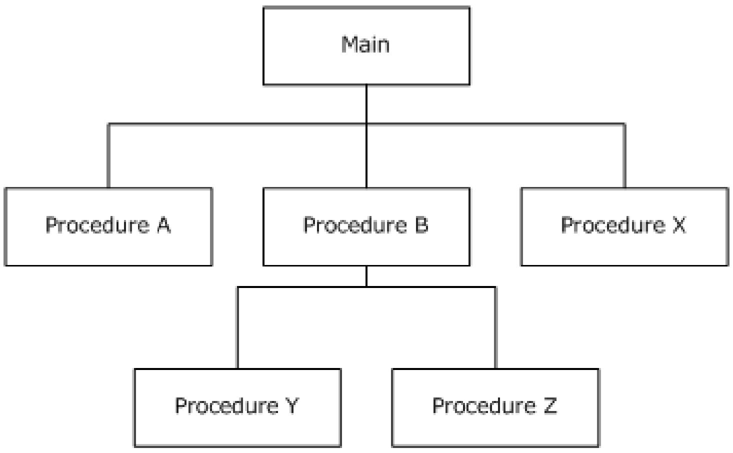


Figure 2: Procedure-oriented code implementation model

Procedure-oriented programming languages provide simple mechanisms for specifying and writing procedures. For example, the ISO-standard C-function prototype is a construct that is used to specify the name of a procedure, the type of the result that it returns (if any), and the number, sequence, and type of its parameters. Using the function prototype is a formal way to specify an interface between procedures.

Microsoft RPC builds on that programming model by allowing procedures that are grouped together in interfaces to reside in different processes than the caller. Microsoft RPC also adds a more formal approach to procedure definition that allows the caller and the called procedure to adopt a contract for remotely exchanging data and invoking functionality. In the Microsoft RPC programming model, traditional function calls are supplemented with two additional elements.

The first element is an .idl/.acf file that precisely describes the data exchange and parameter-passing mechanism between the caller and a called procedure.

The second element is a set of runtime APIs that provide developers with granular control of the remote procedure call, including security aspects, managing state on the server, specifying which clients can communicate with the server, and so on.

3.1.1.1.3 Distributed Systems Model

Traditionally, having software run across multiple computers meant splitting the software into separate client and server components. In such systems, the client component handled the user interface and the server provided back-end processing, such as database access, printing, and so on.

As computers proliferated, dropped in cost, and became connected by ever higher bandwidth networks, splitting software systems into multiple components became more convenient, with each component running on a different computer and performing a specialized function. This approach simplified development, management, administration, and often improved performance and robustness because failure in one computer did not necessarily disable the entire system.

In many cases the system appears to the client as an opaque cloud that performs the necessary operations, even though the distributed system is composed of individual nodes, as illustrated in the following figure.

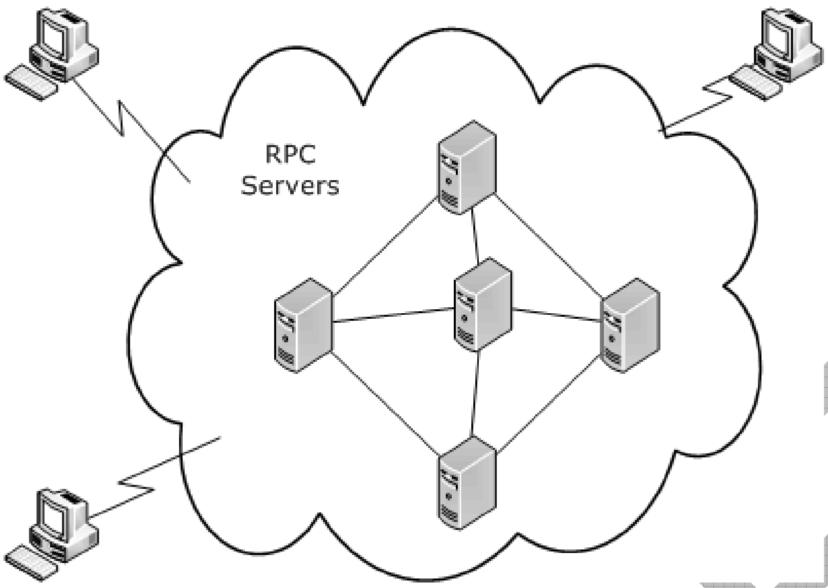


Figure 3: Distributed system paradigm

The opacity of the cloud is maintained because computing operations are invoked on behalf of the client. As such, clients can locate a computer (a *node*) within the cloud and request a particular operation; in performing the operation, that computer can invoke functionality on other computers within the cloud without exposing the additional steps, or the computer on which they were carried out, to the client.

With this paradigm, the mechanics of a distributed, cloud-like system can be broken down into many individual packet exchanges, or conversations between individual nodes.

Traditional client/server systems have two nodes with fixed roles and responsibilities. Modern distributed systems can have more than two nodes, and their roles are often dynamic. In one conversation, a node can be a client; while in another conversation, the node can be the server. In many cases, the ultimate consumer of the exposed functionality is a client with a user sitting at a keyboard and watching the output. In other cases the distributed system functions unattended, performing background operations.

The distributed system may not have dedicated clients and servers for each particular packet exchange, but it is important to remember that there is a caller (or initiator), often referred to as the client. There is also the recipient of the call (often referred to as the server). It is not necessary to have two-way packet exchanges in the request-reply format of a distributed system; often messages are sent only one way.

3.1.1.1.4 How RPC Works

The RPC tools make it appear to users that a client directly calls a procedure that is located in a remote server program. The client and server each have their own address spaces; that is, each has its own memory resource allocated to data that is used by the procedure. The following figure illustrates the RPC architecture.

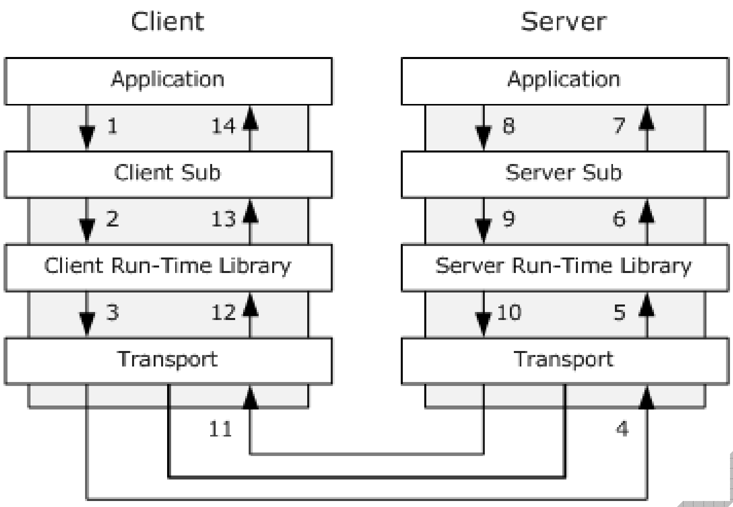


Figure 4: RPC architectural implementation model

As the illustration shows, the client application calls a local stub procedure instead of the actual code that is implementing the procedure. Stubs are compiled and linked with the client application. Instead of containing the actual code that implements the remote procedure, the client stub code:

- Retrieves the required parameters from the client address space.
- Translates the parameters as needed into a standard non-delivery report (NDR) format for transmission over the network.
- Calls functions in the RPC client run-time library to send the request and its parameters to the server.

The server performs the following steps to call the remote procedure.

- The server RPC run-time library functions accept the request and call the server stub procedure.
- The server stub retrieves the parameters from the network buffer and converts them from the network transmission format to the format the server needs.
- The server stub calls the actual procedure on the server.
- The remote procedure then runs, possibly generating output parameters and a return value. When the remote procedure is complete, a similar sequence of steps returns the data to the client.
- The remote procedure returns its data to the server stub.
- The server stub converts output parameters to the format required for transmission over the network and returns them to the RPC run-time library functions.
- The server RPC run-time library functions transmit the data on the network to the client computer.
- The client completes the process by accepting the data over the network and returning it to the calling function.
- The client RPC run-time library receives the remote-procedure return values and returns them to the client stub.

- The client stub converts the data from its NDR to the format used by the client computer. The stub writes data into the client memory and returns the result to the calling program on the client.
- The calling procedure continues as if the procedure had been called on the same computer.

For Windows, the run-time libraries are provided in two parts: an import library, which is linked with the application; and the RPC run-time library, which is implemented as a dynamic-link library (DLL).

The server application contains calls to the server run-time library functions that register the server's interface and allow the server to accept remote procedure calls. The server application also contains the application-specific remote procedures that are called by the client applications.

3.1.1.1.5 Microsoft RPC Components

Microsoft RPC includes the following major components:

- The MIDL compiler.
- Run-time libraries and header files.
- The **name service provider** (sometimes referred to as the locator).
- The **endpoint mapper** (sometimes referred to as the port mapper).

In the RPC model, the **Interface Definition Language (IDL)** is used to formally specify an interface to the remote procedures. The Microsoft implementation of this language is called the Microsoft Interface Definition Language, or MIDL.

If you create an interface using the Microsoft development environment, you must pass it through the MIDL compiler. This compiler generates the stubs that translate local procedure calls into remote procedure calls.

Stubs are placeholder functions that make the calls to the run-time library functions, which manage the remote procedure call. The advantage of this approach is that the network becomes almost completely transparent to your distributed application. Your client program calls what appear to be local procedures; the work of turning them into remote calls is done for you automatically.

All the code that translates data, accesses the network, and retrieves results is generated for you by the MIDL compiler and is invisible to your application.

3.1.1.1.6 Microsoft RPC Security Model

The Microsoft RPC security model builds on the basic Windows security model. MSRPC is network protocol agnostic; the client and server need to agree on a particular protocol, the available protocols are described in [MS-RPCE] section 2.1 Transport. In a similar manner MSRPC is authentication agnostic. A client and server can select from a list of security providers. (SPNEGO, NTLM, Kerberos, Netlogon, No Authentication). The full list is given in [MS-RPCE] section 2.2.1.1.7 Security Providers. The security provider required is indicated by calling the RPC runtime routine RpcServerRegisterAuthInfo (this is the Microsoft implementation of the DCE function rpc_server_register_auth_info). In addition to the choice of security provider there is the concept of authentication level. The authentication level allows fine grain control over the level of protection. The possible authentication levels and the meaning are described in [MS-RPCE] section 2.2.1.1.8 Authentication Levels. There is a detailed example and explanation of a Remote Procedure Services server task in the Authentication Services System Overview [MS-AUTHSO].

3.1.1.2 RPC over HTTP Protocol

The RPC over HTTP protocol [MS-RPCH] specifies the use of either HTTP [RFC2616] or HTTPS [RFC2818] as a transport for the Microsoft Remote Procedure Calls Extensions [MS-RPCE]. In particular, it specifies provisions for using HTTP request/response streams as virtual channels, encoding rules for transporting **RPC PDUs** with HTTP requests and responses, and roles for participants in the protocol.

3.1.1.2.1 Protocol Stack

The following diagram illustrates the protocol stack resulting from the combined roles and encoding rules for the RPC over HTTP protocol [MS-RPCH].

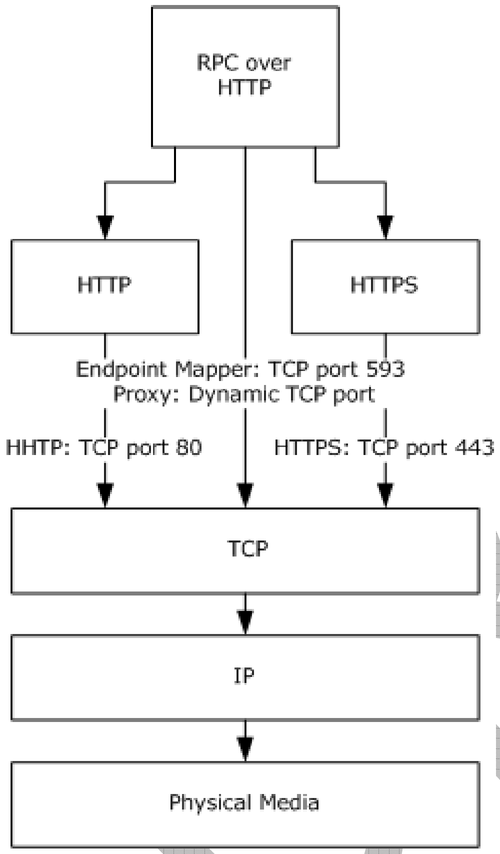


Figure 5: RPC over HTTP protocol stack

As shown in the preceding diagram, the RPC over HTTP protocol is layered directly over the HTTP and HTTPS protocols using their standard TCP port numbers.

The protocol stack for this section does not contain any other protocols.

3.1.1.2.2 Logical Dependencies

The RPC over HTTP protocol [MS-RPCH] is not logically dependent on any protocols other than HTTP, HTTPS, and TCP, which are its underlying transports.

3.1.1.3 Distributed Component Object Model

The Distributed Component Object Model (DCOM) Remote Protocol, as specified in [MS-DCOM], provides mechanisms for exposing application objects in distributed systems via remote procedure calls. It consists of a set of RPC interfaces that may be implemented over any RPC transport by using any RPC protocol sequence that is supported by the Remote Procedure Call Protocol Extensions, as specified in [MS-RPCE]. This layering is illustrated in the following diagram.

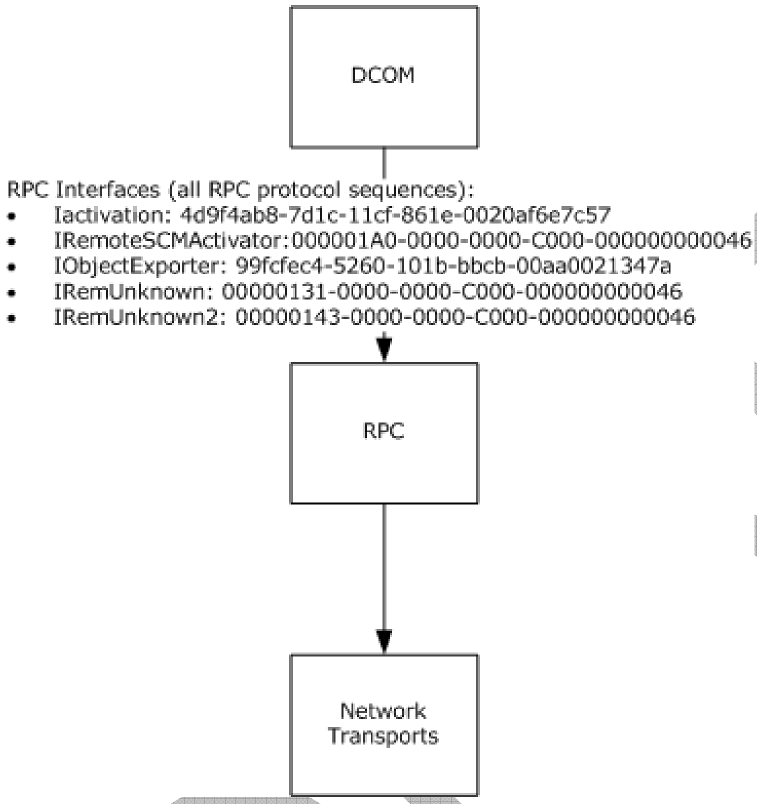

DCOM
RPC Interfaces (all RPC protocol sequences):
• Iactivation: 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
• IRemoteSCMActivator:000001A0-0000-0000-C000-000000000046
• IObjectExporter: 99fcfec4-5260-101b-bbcb-00aa0021347a
• IRemUnknown: 00000131-0000-0000-C000-000000000046
• IRemUnknown2: 00000143-0000-0000-C000-000000000046
RPC
Network Transports

Figure 6: Remote Procedure Call Protocol Extensions stack layers

The DCOM Remote Protocol, as specified in [MS-DCOM], is not logically dependent on any protocols other than the Remote Procedure Call Protocol Extensions.

3.1.1.4 Named Pipes

Many protocols layer on top of named pipes, either directly or indirectly, through the Remote Procedure Call Protocol Extensions as specified in [MS-RPCE]. When named pipes are used, they have the advantage of insulating the higher-layer protocol from what transport was chosen and also of offering the higher-layer protocol the authentication services of the CIFS/SMB/SMB2 connection.

A **named pipe** is a logical connection, similar to a Transmission Control Protocol (TCP) session, between the client and server that are involved in a CIFS/SMB/SMB2connection. The name of the pipe serves as the endpoint for the communication, in the same manner as a port number serves as the endpoint for TCP sessions. This is called a named pipe **endpoint**. The SMB Access Protocols ([MS-CIFS], [MS-SMB], [MS-SMB2] supported by [MS-FSCC] offer the named pipe construct.

A share is a local resource that is offered by an SMB server for access by SMB clients over the network. The protocol defines three types of shares: file (or disk) shares, which represent a directory tree and its included files; pipe shares, which expose access to named pipes; and print shares, which provide access to print resources on the server.

A named pipe share always has the name "IPC$". A named pipe share only allows named pipe operations and Distributed File System (DFS) referral requests to itself. The data that is carried over IPC$ is an implementation detail of SMB as specified in [MS-CIFS]. This implementation detail is transparent to the Remote Procedure Call Protocol Extensions as specified in [MS-RPCE].

3.1.2 Identity and Security Concepts

3.1.2.1 Identity

The first security concept that is important to understand is identity. Identity in Windows comes in several types, and it exists and is managed in several scopes, which are detailed in the following sections. For example, identity can refer to the set of users on a single computer or the identities that are available in a domain.

3.1.2.1.1 Security Principal

A security principal is a common concept in security; it is an actor in a security system and is often something capable of initiating action. Typically, a security principal is associated with a human user of the computer system, but it can also be an autonomous program within the system, such as a logging daemon, system backup program, or something similar.

The security principal is an entity that can be authenticated. In Windows, a security principal is typically a user but can also be a computer or a service. A security principal is often referred to as an *account*.

Security principals receive permissions to access resources such as files and folders. User rights, such as interactive logons, are granted or denied to accounts directly or via membership in a group. The accumulation of these permissions and rights defines what security principals can and cannot do when working on the network.

3.1.2.1.2 Accounts

One of the most important aspects of any security principal is that it serves as a point of management between the system and the administrator. As such, it needs to have attributes that make it meaningful to the administrator or the user. The security principal (or account) has at least a *name* and an *identifier*.

The name is a simple textual name for the account, such as John Smith, SYSTEM, or RedmondDc1$. The name is merely an attribute of the account, however, and can change over time. A common scenario is that the person that the account refers to changes his or her name.

Also, the name is treated as case-*insensitive*. That is, John Smith, JOHN SMITH, john smith, and joHn SmiTH are treated as equivalent in Windows. Microsoft views case-sensitivity as creating an unnecessary burden on the administrator and as something that can lead to mistakes.

The identifier, though also an attribute of the account, has to satisfy other attributes as well. Of particular importance are the uniqueness and persistence of the identifier and the issuer of the identifier. The persistence of the identifier is what provides the administrator with the capability to assign a resource to that account and not be surprised in the future by changes to the account.

Consider the case of John Smith. The administrator may assign John Smith access to a certain document at one point in time. If that John Smith leaves the company and a new John Smith is hired, the new John Smith should not have access to the resources of the original John Smith. Conversely, if John Smith changes his name to John Q. Smith, he should not lose access to the resources previously granted.

The other important attribute is the issuer of the identifier. Identities have different weight, conceptually, depending on the issuer. In the physical world, a store is generally willing to accept a driver's license as proof of identity, but the store is unwilling to accept a gymnasium membership card. In the Windows model, the issuer of an account is encoded with the identity so that any recipient can make a similar decision.

The identifier that Windows uses for accounts is called a *security identifier* (*SID)*.

Windows contains a number of built-in accounts:

- **User account**: Identifies users who belong to the domain by storing their names, their passwords, the groups that they belong to, the permissions that they have for accessing system resources, and other personal information.
- **Group account**: Identifies a specific group of users and is used to assign them permissions to objects and resources.
- **Computer account**: Identifies computers that belong to the domain. A computer account is commonly referred to as a "machine account."

Each built-in user, computer, or group account is a security principal.<1>

User and computer accounts can be added, disabled, reset, and deleted by using Active Directory Users and Computers. A computer account can also be created when a computer is joined to a domain. For more information about user and computer accounts, see Active Directory naming and Object names.

3.1.2.1.3 User accounts

In the Active Directory directory service, each user account has a user logon name, a pre-Microsoft Windows 2000 user logon name (security accounts manager (SAM) account name), and a user principal name (UPN) suffix. The administrator enters the user logon name and selects the UPN suffix when creating the user account. Active Directory suggests a pre-Windows 2000 user logon name that uses the first 20 bytes of the user logon name. Administrators can change the pre-Windows 2000 logon name at any time.

In Active Directory, each user account has a UPN based on [RFC822]. The UPN is composed of the user logon name and the UPN suffix joined by the "at" sign (@).

Note When creating a user account, it is not necessary to add the at sign (@) to the user logon name or to the UPN suffix. Active Directory automatically adds the at sign (@) when it creates the UPN. A UPN that contains more than one at sign (@) is invalid. User logon names do not end with a period or multiple periods.<2>

The second part of the UPN, the UPN suffix, identifies the domain in which the user account is located. This UPN suffix can be the Domain Name System (DNS) domain name, the DNS name of any domain in the forest, or an alternative name that is created by an administrator and used just for logon purposes. This alternative UPN suffix does not need to be a valid DNS name.

In Active Directory, the default UPN suffix is the DNS name of the domain in which the user account is created. In most cases, this is the domain name that is registered as the enterprise domain on the Internet. Using alternative domain names as the UPN suffix can provide additional logon security and simplify the names that are used to log on to another domain in the forest.

For example, if an organization uses a deep domain tree that is organized by department and region, domain names can become quite long. The default UPN for a user in that domain might be sales.westcoast.contoso.com. The logon name for a user in that domain would be user@sales.westcoast.contoso.com. Creating a UPN suffix of "contoso" would allow that same user to log on by using the much simpler logon name of user@contoso.

3.1.2.1.4 Computer Accounts

Each computer account that is created in Active Directory has a relative distinguished name (RDN), a pre-Microsoft Windows 2000 computer name (SAM account name), a primary DNS suffix, a DNS host name, and a service principal name (SPN) in addition to the computer name. The administrator enters the computer name when creating the computer account. The computer name must include the dollar sign ($) character at the end of the name (for example RedmondDc1$).

When the domain functional level has been set to Windows Server 2003 operating system, a new **lastLogonTimestamp** attribute is used to track the last logon time of a user or computer account. This attribute is replicated in the domain and can provide important information regarding the history of a user or a computer.

Every Windows computer that joins a domain has a computer account.<3> Similar to user accounts, computer accounts provide a means for authenticating and auditing computer access to the network and to domain resources. Each computer account must be unique.

When the Netlogon service running on a client computer connects to the Netlogon service on a domain controller (DC) in order to authenticate a user, the Netlogon services challenge each other to determine whether they both have a valid computer account. This allows a secure communication channel to be established for logon purposes.

In order for a Windows computer to join a domain, the computer must have a computer account in Active Directory.<4>

The computer name (for example RedmondDc1$) is used as the Lightweight Directory Access Protocol (LDAP) relative distinguished name (RDN). Active Directory suggests the pre-Windows 2000 name that uses the first 15 bytes of the RDN. The administrator can change the pre-Windows 2000 name at any time.

The DNS name for a host is called a full computer name and is a DNS fully qualified domain name (FQDN). The full computer name is a concatenation of the computer name (the first 15 bytes of the SAM account name of the computer account without the "$" character) and the primary DNS suffix (the DNS domain name of the domain in which the computer account exists). It is listed on the Computer Name tab in the System Properties dialog box in Control Panel.

By default, the primary DNS suffix portion of the FQDN for a computer must be the same as the name of the Active Directory domain where the computer is located. To allow different primary DNS suffixes, a domain administrator may create a restricted list of allowed suffixes by creating the **msDS-AllowedDNSSuffixes** attribute in the domain object container. This attribute is created and managed by the domain administrator by using Active Directory Service Interfaces (ADSI) or LDAP.

The SPN is a multivalue attribute. It is usually built from the DNS name of the host. The SPN is used in the process of mutual authentication between the client and the server hosting a particular service. The client finds a computer account based on the SPN of the service to which it is trying to connect. The SPN can be modified by members of the Domain Admins group.

3.1.2.1.5 Security Identifiers (SIDs)

The security identifier (SID) is an account identifier. The SID is variable in length and encapsulates the hierarchical notion of issuer and identifier. It consists of a 6-byte *identifier authority* field that is followed by one to fourteen 32-bit *subauthority* values and ends in a single 32-bit *relative identifier (RID)*. For example, a two-subauthority SID appears as shown in the following figure.

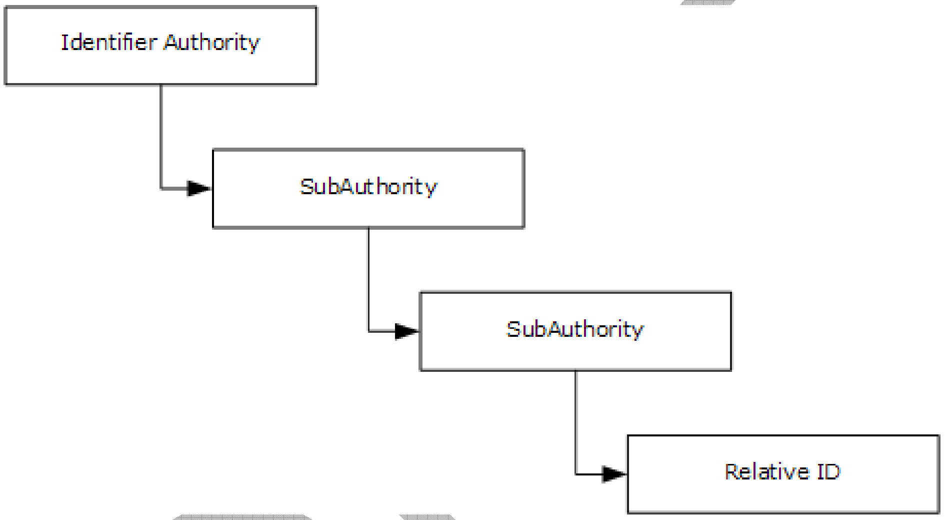


Figure 7: Windows security session identifier via subauthorities

When displayed textually, the accepted form is the following:

S-1-<identifier authority>-<sub1>-<sub2>-…-<subn>-<rid>

where *S* and *1* are literal strings, *identifier authority* is the 6-byte value, *sub1* through *subn* are the subauthority values, and *rid* is the RID.

The original concept of the SID called out each level of the hierarchy. Each layer included a new subauthority, and an enterprise could lay out arbitrarily complicated hierarchies of issuing authorities. Each layer could, in turn, create additional authorities beneath it. In reality, this system created a lot of overhead for setup and deployment and made the management model group even more complicated. The notion of arbitrary depth identities did not survive the early stages of Windows® development, although the structure was already too deeply ingrained to be removed.

In practice, two SID patterns developed. For built-in, predefined identities, the hierarchy was compressed to a depth of two or three subauthorities. For real identities of other principals, the identifier authority was set to five, and the set of subauthorities was set to four.

Whenever a new issuing authority under Windows is created (for example, a new machine deployed or a domain created), it is assigned a SID with 5 (an arbitrary value) as the *identifier authority*; a fixed value of 21 is used as a unique value to root this set of subauthorities, and a 96-bit random number is created and parceled out to the three subauthorities with each subauthority that receives a 32-bit chunk.

Windows allocates RIDs starting at 1,000; RIDs having a value less than 1,000 are considered reserved and are used for special accounts. For example, all Windows accounts with a RID of 500 are considered built-in Administrator accounts in their respective issuing authorities.

Thus, a SID that is associated with an account appears as depicted in the following figure.

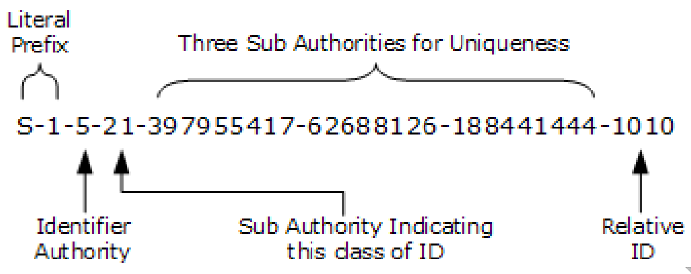

Literal Prefix
Three Sub Authorities for Uniqueness
S-1-5-21-397955417-62688126-188441444-1010
Identifier Authority
Sub Authority Indicating this class of ID
Relative ID

Figure 8: SID with account association

For most uses, the SID can be treated as a single long identifier for an account. By the time a specific SID is associated with a resource or logged in a file, it is effectively just a single entity. For some cases, however, it should conceptually be treated as two values: a value that indicates the issuing authority and an identifier relative to that authority. Sending a series of SIDs, all from the same issuer, is one example: the list can easily be compressed to be the issuer portion and the list of IDs relative to that issuer.

It is the responsibility of the issuing authority to preserve the uniqueness of the SIDs, which implies that the issuer must not issue the same RID more than one time. A trivial approach to this entails allocating RIDs sequentially. More complicated schemes are certainly possible; Active Directory uses a multimaster approach that allocates RIDs in blocks. It is possible for an issuing authority to run out of RIDs; therefore, the issuing authority must take care to handle this situation correctly. Typically, that authority must be retired.

Windows supports the concept of groups with much the same mechanisms as individual accounts. Each group has a name, just as the accounts have names. Each group also has an associated SID.

User accounts and groups share the same SID and namespaces. Users and groups cannot have the same name on a Windows-based system nor can the SID for a group and a user be the same.

For access control, Windows makes no distinction whether a SID is assigned to a group or an account. Changing the name of a user, computer, or domain does not change the underlying SID for that account, and an administrator cannot modify the SID for that account. Administrators cannot modify the SID for an account in Microsoft Windows® NT, and there is generally no need to know the SID that is assigned to a particular account. SIDs are primarily intended to be used internally by the operating system to ensure that accounts are uniquely identified in the system.

3.1.2.1.6 Groups

A group is a collection of user accounts, computer accounts, and other groups that can be managed as a single unit from a security perspective. Groups can either be based in Active Directory or local to a particular computer.

Windows Server has several built-in accounts and security groups that are preconfigured with the appropriate rights and permissions to perform specific tasks.<5> Active Directory provides two types of administrative responsibility: service administrators are responsible for maintaining and delivering the directory service, including DC management and directory service configuration; data administrators are responsible for maintaining the data that is stored in the directory service and on servers and workstations that are domain members.

It is important to understand which default accounts and groups are service administrators. Service administration accounts and groups have the most widespread power in the network environment and require the most protection.

3.1.2.1.6.1 Group Types

Starting with Windows 2000, Windows provides two types of groups:

- Security groups: These groups can contain members and can be granted permissions in order to control access to network resources. Security groups can contain users, other groups, and even computers.

- Distribution groups: These groups are used for nonsecurity functions, such as grouping users together to send e-mail messages. Unlike security groups, these groups cannot be used to control access to network resources.

Microsoft Windows NT Server has only one type of group, which is equivalent to security groups in Microsoft Windows 2000 Server operating system, Windows Server 2003 operating system, and Windows Server 2008 operating system.

Starting with Windows 2000 Server, groups can be created by using the Active Directory Users and Computers console, and groups are stored as group objects in Active Directory. In Windows NT, groups can be created by using User Manager for Domains, and groups are stored in the SAM database. Users can belong to multiple groups at the same time. A group does not actually contain its member user accounts; it is just a list of user accounts.

3.1.2.1.6.2 Group Scope

The scope of a group can be local or global depending on the portion of the network for which the group can be granted rights and permissions. Beginning with Windows 2000, Windows provides three levels of scope for security groups:

- Universal groups: These groups can contain members for any domain and be granted permissions to resources in any domain in a specific Active Directory forest. An Active Directory forest is a collection of one or more Active Directory domains that share a common logical structure, directory schema, and network configuration, as well as automatic two-way transitive trust relationships. Each forest is a single instance of the directory and defines a security boundary. Universal groups can contain user accounts, global groups, and universal groups from any domain in the current forest. An administrator can create a universal group only when the domain is in native mode and not in mixed mode as defined in section 3.1.2.1.6.3.

- Global groups: These groups can contain members only from their own domain but can be granted permissions to resources in any trusting domain. When the domain is in native mode, global groups can contain user accounts and global groups from the same domain. When the domain is in mixed mode, these groups can contain only user accounts.

- Domain local groups: These groups can contain members from any trusted domain but can be granted permissions only to resources in their own domain. Unlike Windows NT local groups, a domain local group can be granted permissions to resources on all servers (both the DCs and member servers) in its domain. When the domain is in mixed mode, domain local groups can contain user accounts and global groups from any trusted domain or forest. When the domain is in native mode, domain local groups can also contain domain local groups from their own domain and universal groups from within any domain in the forest.

Beginning with the Windows 2000 Server operating system, for member servers and client computers, and also for Windows XP operating system clients, a fourth scope of group called a local group can exist only within the local security database of the computer where it is created. Local groups in Windows 2000 Server, Windows Server 2003 operating system, Windows Server 2008 operating system, Windows XP, and Windows Vista operating system are similar to local groups in Windows NT. They can contain user accounts that are local to the computer and user accounts and global groups from their own domain. A local group can be granted permissions to resources only on the computer where it was created. The Local Users and Groups Microsoft Management Console (MMC) is used to create local groups on a computer.

Windows NT groups have only two levels of scope:

- Global groups: A global group can be granted permissions to resources in its own domain and to resources in trusting domains. A global group can contain user accounts only from its own domain. Global groups are created on Windows NT domain controllers and exist in the domain directory database.
- Local groups: A local group created with Windows NT Workstation can be granted permissions only to resources on the computer where it was created. A local group created with Windows NT Server domain controller can be granted permissions only to resources on the domain controllers of its own domain. A local group can contain user accounts and global groups both from its own domain and from trusted domains. Network administrators of enterprise-level Windows NT networks can use a resource-access strategy called AGLP (Accounts organized by placing them in Global groups, which are then placed in Local groups that have appropriate Permissions and rights assigned to them) to plan and implement local groups in their network.

Beginning with Windows 2000 Server, the scope of a group can be changed. For example, global groups that are not members of other global groups can be converted to universal groups. Domain local groups that do not contain other domain local groups can be converted to universal groups.

3.1.2.1.6.3 Nested Groups

Windows supports the concept of nesting groups, or adding groups to other groups. Nesting groups can help reduce the number of permissions that need to be individually assigned to users or groups.

The process of creating groups across domains involves the following steps:

1. The administrator in each domain creates global groups and adds user accounts that have the same resource requirements to the global groups.
2. A domain administrator creates a domain local group for each resource that exists within a domain, such as file shares or printers, and then adds the appropriate global groups from each domain to this domain local group.
3. A domain administrator assigns the appropriate permissions for the resources to the domain local group. Users in each global group receive the required permissions because their global group is a member of the domain local group.

Effectively nesting groups in a multidomain environment reduces network traffic between domains and simplifies administration in a domain tree. A domain tree is a collection of domains that are grouped together in hierarchical structures so that they can be administered as single logical unit.

When a domain is added to the domain tree, it becomes a child of the tree root domain. The domain to which a child domain is attached is called the parent domain. A child domain can contain its own child domain. The name of a child domain is combined with the name of its parent to form its own unique DNS name such as Corp.mycompany.msft. In this manner, a tree has a contiguous namespace.

The extent to which nesting can be used in a specific organization depends on which mode the DC was configured in the system. Domain controllers can be configured in two modes: mixed mode or native mode.

- **Mixed mode**: A DC that is configured to support a mixed environment with Windows NT 4.0 operating system, Windows 2000, Windows Server 2003 operating system, and Windows Server 2008 operating system domain controllers in the same domain.
- **Native mode**: A DC that is configured to support only mixed Microsoft Windows 2000 Server operating system, Windows Server 2003, and Windows Server 2008 environments.

In mixed mode, only one type of nesting is available; global groups can be members of domain local groups. Universal groups do not exist in mixed mode. In native mode, multiple levels of nesting are available. The nesting rules for group memberships for Windows 2000 are summarized in the following table.

Group scope	Can contain	Can be a member of
Domain local group	User accounts and universal and global groups from any trusted domain. Domain local groups from the same domain.	Domain local groups in the same domain.
Global group	User accounts and global groups from the same domain.	Universal and domain local groups in any domain. Global groups in the same domain.
Universal group	User accounts and universal and global groups from any domain.	Universal or domain local groups in any domain.

3.1.2.1.7 Account Domains

Accounts are always created relative to some issuing authority, which is responsible for allocating and assigning the SID. For Windows issuing authorities, this is referred to as a *domain*. Domains come in two varieties, local and remote.

3.1.2.1.7.1 Local Domains

Every computer running Windows has a local domain; that is, it has an account database for accounts that are specific to that computer. Conceptually, this is an account database like any other with accounts, groups, SIDs, and so on. These are referred to as local accounts, local groups, and so on. Because computers typically do not trust each other for account information, these identities stay local to the computer on which they were created.

3.1.2.1.7.2 Remote Domains and Domain Controllers

With a remote domain, certain Windows-based servers can be configured to be *DCs*. A DC is a server that has made its account database available to other machines in a controlled manner. Starting with Windows 2000,<6> DCs began supporting a database of more than just accounts, becoming a general-purpose directory. This is known as Active Directory.

Because the account database is typically distributed across multiple DCs, there can naturally be a mix of different versions of the individual servers. Active Directory has the notion of a *functional level*, which serves as a version level for the entire directory. The functional level is managed by the administrator and the system itself.

A domain has built-in groups; these groups are defined by Microsoft and created within the domain during installation. For example, built-in groups include the Domain Users, Domain Computers, and Domain Admins groups. By default, the Domain Users group includes all users who are defined in the domain.

A DC accepts authentication requests on behalf of the machines that have chosen to trust it.

A DC can have peers within the domain. These peers are other servers that also have been configured to host this account database. Any server participating in the domain as a domain controller may or may not allow changes; the configuration is a choice of the administrator.<7>

When a change is allowed, the servers replicate the change so that all DCs have the same information.

3.1.2.1.7.3 Domain Membership

Domain membership is the state of trusting a third party (the DC) for identity and authentication information. Any system can conceivably be part of a domain. Windows-based systems can easily be configured to be part of a domain and trust their DC for many tasks. Also, certain configuration changes are made, such as accepting the domain as the authoritative source of time.

Windows-based systems can have local groups that include members from a domain. This allows the member system to manage its resources in the manner most relevant to it and not be completely dependent on the decisions of the domain administrator. A domain administrator can create a domain local group for each resource that exists within a domain, such as file shares or printers, and then add the appropriate global groups from each domain to this domain local group. The domain administrator then assigns the appropriate permissions for the resources to the domain local group.

Joining a domain ultimately distills down to establishing an account on the domain that represents the system joining the domain, and to setting the password (or key) for the account on both the domain and the actual system. In Windows, this process is encapsulated in a domain join function (NetJoinDomain). Several tools, such as WinBind, exist for non-Windows operating systems to join a Windows domain.

All Windows-based systems have a component that manages their relationship with their DC. This component, called *Netlogon*, maintains the keys that are necessary for ongoing authentication of the member system to the DC. It also creates a general-purpose channel to the Netlogon instance on the DC.<8>

This channel is used by various authentication protocol implementations to redirect an authentication request to (or augment their activities with) their instance on the DC.

3.1.2.1.7.4 Effect on Accounts

Windows domains have an effect on the way that accounts and groups work. Some of this is by convention, and some is by design.

By convention, when a Windows-based system is added to a domain, the domain administrators group is made a member of the local administrators group.

By design, groups have different scopes when domains are involved. Groups can be defined to be globally known and therefore usable by other domains or known only within the domain in which they are defined.

3.1.2.1.8 Globally Unique Identifiers (GUIDs)

In Windows programming and in Windows operating systems, a globally unique identifier (GUID), as specified in [RFC4122], is a 128-bit value that is a binary unique identifier (ID) for a specific entity. The term universally unique identifier (UUID) is sometimes used in Windows protocol specifications as a synonym for GUID.

3.1.2.1.8.1 Uniqueness

All GUIDs are assumed to be unique; however, it cannot be said that they MUST be unique because there is no mechanism to enforce that uniqueness. Some GUIDs are also unpredictable. [RFC4122] defines five versions of GUID, one of which, version 4, is unpredictable by design. Because a GUID includes a version number field, no GUID of one version could equal a GUID of a different version.

Sometimes GUIDs are generated at design time and remain constant throughout the life of a protocol, such as a GUID that identifies a remote procedure call (RPC) interface or one that identifies a particular Active Directory schema. Such GUID values when used in a protocol are typically listed in the protocol document. Other GUIDs are generated at runtime by the protocol implementation itself and are used to identify transitory things such as individual sessions, connections, transactions, and activities.

Some protocols use unpredictable GUIDs as self-authenticating identifiers or nonces. That is, the GUID value (for example, a GUID that represents a client ID) is kept secret by both parties to the protocol and is used as an identifier. However, because it is assumed to have significant entropy, it also serves as a high-entropy password. Alternatively, a random GUID can be used as a nonce, which is a number that is used only one time and is unpredictable by the attacker. A nonce is typically used for the purpose of preventing replay attacks.

3.1.2.1.8.2 Internal Structure

A GUID has substructure, which a GUID generator needs to be aware of. A protocol stack implementation that uses a GUID, unless the individual protocol specification explicitly states the contrary, SHOULD treat that GUID as an opaque single quantity to which only equality or inequality tests are applied. Such a protocol implementation SHOULD NOT make decisions based upon the substructure of the GUID.

[RFC4122] defines five versions of GUID and specifies how they are constructed. The known substructure is represented by the version and variant fields. The remaining fields (what that RFC calls time stamp and node) are what make the GUID unique.

In a version 1 GUID, these fields carry a value that is derived from the time and the computer's network node address, respectively. A side effect of a version 1 GUID is that it identifies the machine on which it was generated (barring replacement or transfer of its network interface card).

In a version 4 GUID, these fields have been replaced by random bits. In generating a version 4 GUID, an implementation SHOULD use a Federal Information Processing Standard (FIPS)-approved pseudo-random number generator (PRNG) (as specified in [FIPS140-2] or later), but any superior source of random bits (such as a true hardware PRNG) MAY be used instead. For more information about entropy sources, see [RFC4086]. If a PRNG is used as the source of random bits, it takes a parameter called a *seed*. Although many experts in the field have studied, tested, and approved each FIPS-approved PRNG algorithm, no approved or nonapproved PRNG output is or can be more unique or less guessable than its seed.<9><10>

Any implementation of version 4 GUIDs MAY implement two types of GUID: one GUID for uniqueness only and another GUID for use as a nonce. These types are characterized as follows:

- **Uniqueness-only**: This GUID is not kept secret and it is not required that it cannot be guessed. Therefore, the PRNG that is used to generate a version 4 GUID needs to be seeded only with values that have a high probability of being unique for that machine and that session. Typically, such a seed is formed by the cryptographic hash of uniqueness values, such as the CPU ID, MAC address, time of day, processor tick count, system process table, and system state.

- **Nonce or authenticator**: This GUID must be kept secret to preserve its security value. The PRNG that is used to generate a version 4 GUID for this purpose needs to be seeded with values that have a high probability not only of being unique but also of being unguessable by an attacker. Typically, such a seed is formed by the cryptographic hash of unguessable values of high entropy, such as the output of a hardware True Random Bit Generator, the system state readable in kernel mode, the history of system state over a long runtime, the accumulated entropy from earlier operation of the system (retained in a place that the attacker cannot access), the time of arrival of hardware interrupts, or the history of mouse positions as it moves.

A secret GUID that is good enough for a nonce is also good for uniqueness. Any system that generates cryptographic keys needs a source of true random or pseudo-random bits that are unguessable enough for building those keys. Therefore, it is common practice for an implementer to provide only one type of version 4 GUID, which is a type that is good enough for a nonce; and to use cryptographic-quality random bits for generating that GUID—even when building a GUID that is used only for uniqueness.<11>

3.1.2.1.8.3 Quality of Random Bits

Each use of a version 4 GUID has a measure of quality:

- **Uniqueness**: The probability that some other system will happen to generate the same GUID (without any conscious attempt by an attacker to create that collision).
- **Nonce**: The probability that some conscious attacker will be able to guess the generated GUID.

An ideal GUID that uses N bits of randomness collides with some chosen value by accident (violating uniqueness) with a probability proportional to 2^{-N} and is guessable by an attacker with work proportional to 2^{N}. This quality of randomness is measured by entropy. The ideal case that is described earlier is said to represent N bits of entropy.

If N allegedly random bits actually contain M<N bits of entropy, the probability of accidental collision is proportional to 2^{-M} and the work for an attacker is proportional to 2^{M}. This leads, in the case of GUIDs for uniqueness, to a higher-than-ideal probability of accidental collision. If such a collision occurs, the two different identified objects will have the same ID, possibly leading to confusion. In the case of GUIDs for use as nonces, the lower work by the attacker might result in a successful replay attack.

Neither of these flaws, if one occurs, changes any protocol that uses such a GUID. It might change the security claims that such a protocol might make, but not the state machine, packet sequence, or packet contents of the protocol. The same applies to GUIDs that are used for uniqueness. If two GUIDs that were supposed to be different are accidentally the same, then the protocol implementation is not changed, only the probability that a mistaken identity might occur.

Therefore, the implementer of a protocol is not required by the protocol specification to guarantee any quality of random bits. Nothing in the specification of any Microsoft Communications Protocol Program (MCPP) protocol directs a conformant implementation to look for, much less detect, any use of low-quality random bits in GUID generation. This means that even a very low-entropy random bit stream can be used to generate GUIDs that will allow a protocol to interoperate and be indistinguishable from any other protocol implementation, no matter what the quality of random bits in that other implementation.

Of course, implementers who prefer to minimize the confusion that would result from nonunique GUIDs or the replay attacks that would result from guessable nonces are well advised to use the best-quality random bit sources that they can find.

3.1.2.2 Authentication

This section describes how identity is proven across a Windows network.

The purpose of authentication is for two communicating entities to establish the identity of one or both parties. It is presumed that the communication medium between the two entities is completely hostile, and that an attacker can inspect any message or tamper with any message. Tamper here means change, suppress, or replay. Protocols must be developed that allow the two entities to authenticate in such a harsh environment. Commonly, the two entities consist of a client and a server.

Authentication can take on several aspects. For example, authentication of the server may be sufficient. The use of Secure Sockets Layer (SSL) on the Internet is primarily centered on assuring the client of the identity of the server. For protected networks, client authentication may be sufficient, because the valuable resource lives on a single server, and the server really only needs to address the identity of the client.

On modern networks, however, proving the identity of both the client and the server has become of the highest importance. Clients need to be assured of the identity of a server to avoid divulging something important to a rogue server; servers must be assured of the identity of clients to avoid granting clients inappropriate access. This security concept is typically called *mutual authentication*.

Ultimately, authentication must be performed by using cryptographic operations of some form, such as encryption or signatures. There are two main types of encryption: *symmetric* and *asymmetric*. Symmetric encryption uses the same key to encrypt and decrypt a message. Asymmetric encryption uses one key to encrypt and uses a different key to decrypt; these keys are linked by mathematical requirements. Signatures can be implemented in a number of ways through keyed hashes and encrypted hashes.

In the early 1990s, John Linn, then of Digital Equipment Corporation, proposed that applications not be tied to specific security protocols. This proposal was the genesis of the Generic Security Service Application Programming Interface (GSS-API). This concept has driven the model of most authentication protocols that are intended for use within an application protocol. This concept is generally referred to as GSS style or the GSS model. Note that there have been a number of channel-based protocols, such as Secure Sockets Layer (SSL), Transport Layer Security (TLS), and Secure Shell (SSH), that are intended to be below the application protocol layer.

This approach, however, has led to a simplified form of interaction between the application protocol and the authentication protocol. In this model, the application protocol is responsible for ferrying discrete, opaque packets that are produced by the authentication protocol. The application has no visibility into the contents of the message; its responsibility is merely to carry them. These messages, which GSS specifications refer to as *tokens*, implement the authentication process.

The application in this model first calls the authentication protocol on the client. The client portion of the authentication protocol creates a token and returns it to the calling application. The application then transmits that token to the server side of its connection, embedded within the application protocol. On the server side, the server application extracts the token and supplies it to the authentication protocol on the server side. The server-side authentication protocol can process the token and possibly generate a response or determine that authentication is complete. If another token is generated, the application must carry it back to the client, where the process continues.

This exchange of security tokens continues until one or both sides determine that authentication is complete. If authentication fails, the application should drop the connection and indicate the error. If it succeeds, the application can then be assured of the identity of the participants, as far as the underlying protocol can accomplish.

When authentication is complete, session-specific security services can be available. The application can then invoke the authentication protocol to sign or encrypt the messages that are sent as part of the application protocol. These operations are done in much the same way, where the application can indicate what portion of the message is to be encrypted, and then must include a per-message security token. By signing or encrypting the messages, or both, the application can obtain privacy, resistance to tampering of messages, and detection of messages dropped, suppressed, or replayed.

Windows networking has its roots in the LAN Manager (LM) network product. LM was designed for a time when client authentication was sufficient for most needs, and when computational capacity was exceeded by the algorithms common at the time. For example, exhaustively searching Data Encryption Standard (DES) keys was unthinkable by any but the most dedicated government resources. LM authentication used a straightforward challenge-response style of authentication and was sufficient for many customers for many years.

When Microsoft decided to adopt the Kerberos protocol for Windows and move away from NT LAN Manager (NTLM), it required a substantial change for a number of protocols. This process is still going on today. Rather than repeat the process when circumstances required a new or additional security protocol, Microsoft chose to insert a protocol, in this case, Simple and Protected GSS-API Negotiation (SPNEGO), to allow security protocol selection and extension.

3.1.2.2.1 Authentication Protocols

Several protocols are available for authentication in Windows, each with different strengths and weaknesses, different capabilities, and different uses within the product.

3.1.2.2.1.1 NT LAN Manager (NTLM)

NT LAN Manager (NTLM) is an ongoing extension to the original LAN Manager (LM) authentication protocol. NTLM is conceptually straightforward and only performs client authentication. NTLM has undergone some revision (known as NTLMv2), which incorporates additional information into the computation of the response; however, it still follows the same general message flow.

NTLM can never provide mutual authentication in all situations; the client can never be assured of the identity of the server in a general manner. The only time that NTLM can provide mutual authentication is when the client knows through out-of-band means that the user name used for authentication exists only on the target server. This can constrain the authentication to at least within the extended set of trusted domains.

3.1.2.2.1.2 Kerberos

In 1993, Microsoft began working toward adopting the Kerberos protocol. Kerberos support for mutual authentication, transitive trust among authorities, extensibility, public inspection and review, and performance and caching, all pointed to this protocol as the authentication protocol for enterprise deployment for the foreseeable future. Microsoft Windows® 2000 shipped in 1999 with the Kerberos protocol natively supported.

Because the Kerberos protocol is an Internet Engineering Task Force (IETF) protocol, a multitude of documents are available that describe how it works; see [RFC4120] as the canonical reference.<12>

MIT also has a document that describes the ideas behind the Kerberos protocol in an approachable manner (see Designing an Authentication System: A Dialogue in Four Scenes [DIALOGUE]).

Kerberos is strictly an authentication protocol; it is designed to convey only the identity of the principals on each side of the connection. The Kerberos protocol does contain extensibility points to allow extra vendor-defined information to be conveyed (along with the ticket) during authentication. This is expressed as the *auth_data* field in the Kerberos ticket.

Windows uses this *auth_data* field in the ticket to pass along the SID of the account and to group SIDs during authentication. The structure that is used for this behavior is termed the privilege attribute certificate (PAC). The PAC contains all the account's group memberships and is used to provide sufficient context on the server to make authorization decisions (for example, "can this user open this file?"). The encoding of the group information is publicly available from Microsoft. See Utilizing the Windows 2000 Authorization Data in Kerberos Tickets for Access Control to Resources [KERBPAC].

Although group membership information is a common use of the *auth_data* field, Microsoft also included additional profile and account management information, such as the location of the account's root directory. This design greatly aids certain enterprise deployment scenarios. This additional information is used for supporting interactive logon. These additions are available through the Microsoft Communications Protocol Program [MSFT-LEGALMCPP].

A strength of the Kerberos protocol is that it is not involved after the ticket is issued until the ticket expires and the client needs a new one. This behavior means that the client and server do not need to involve the KDC for every connection.

Much like other protocols, the Kerberos protocol can leverage more than one domain. If the server that is requested by a client is in a different domain, the KDC can return a TGT to that other domain. The client can then retry the request on a KDC in the other domain. This can span multiple domains as long as a trusting relationship exists among the domains.

One issue with a secret key system such as the Kerberos protocol is that any person who knows the key for a principal can create a ticket to that principal. Thus, even though the KDC is the normal creator of tickets, a person who knows the password for a principal could create and send a ticket to that principle.

For authentication only, this action is not a threat. The ticket created by the person is, in effect, fooling the principal with this action.

When the ticket includes authorization data that is respected and interpreted by something other than code running as that user, the potential for problems arises. In the Windows implementation, the authorization data results in a Windows *access token*, which is a system-provided object that encapsulates an account's identity, group memberships, and system privileges. An access token on Windows is used to make authorization decisions by the system (for example, validating access to a file).

A malicious user might construct a ticket to itself that contains a PAC indicating that the user is a member of a group (for example, the Administrators group) that it should not be. If processed naively, the system accepts this PAC and allows the user inappropriate access to the system. Windows compensates for this by involving the DC when a nonprivileged server receives a ticket. The Windows Kerberos implementation calls back to the DC through the Netlogon channel to verify the contents of the PAC.

3.1.2.2.1.3 Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

SSL is a protocol first introduced by Netscape to allow browsers to authenticate servers. SSL went through a number of revisions, culminating in an adoption by the IETF and becoming the TLS protocol. For more information, see [RFC2246].

Although SSL and TLS are primarily used for authenticating servers and creating a secure pipe between the client and the server, they do allow for authenticating the client. At almost any point after the channel is established, the server can demand client authentication. The client signs a challenge from the server with its private, asymmetric key, and then sends both the signed data and the certificate for that key to the server.

In Windows®, the certificate can be associated with an account in several ways, based on local policy and on what fields from the certificate are interesting to the administrator of the server or domain. Ultimately, however, the SSL/TLS implementation on the server calls up to the DC through the Netlogon channel and asks the SSL instance on the DC to determine what account is associated with this certificate.

The account is determined through a number of possible mappings, based on the fields present in the certificate. For example, *subjectAltName*, *commonName*, and other similar fields can be used to find the account in Active Directory. After the account is found, the account information (such as account SID and group membership SIDs) is returned to the client. Much like the NTLM case involving the domain, this can extend through arbitrary trust relationships. The format for this information is the same as the format of the PAC data from the Kerberos protocol.

3.1.2.2.1.4 Simple and Protected GSS-API Negotiation Mechanism (SPNEGO)

SPNEGO is an authentication protocol that actually does no authentication. Rather, it is an authentication protocol that allows secure negotiation among other security protocols when the client and the server might support more than one protocol. Windows® uses SPNEGO instead of a specific protocol to allow for simpler substitutions of additional security protocols.

In Windows, SPNEGO has the important task of deciding which protocol is valid for a particular connection. It does this by interpreting a number of input states and then adjusting the list of security protocols it offers the server. The SPNEGO implementation knows that NTLM cannot implement the same guarantees (due to legacy) that more modern protocols can. Therefore, NTLM is treated in a special manner on a Windows-based system. To decide the most appropriate protocol, the SPNEGO implementation takes the following information:

- User domain functional level, if known
- Systemwide mutual authentication policy
- Name of the server

If the user is a member of a Windows 2000 or later domain, SPNEGO knows that NTLM is a second choice and that at least the Kerberos protocol should be available. Therefore, if the Kerberos protocol indicates that it should be used (by returning an optimistic token), SPNEGO does not allow further downgrade to NTLM if the Kerberos authentication fails. Also, if the KDC cannot be contacted, and the user is a member of a Windows 2000 or later domain, the SPNEGO implementation fails, indicating that it cannot authenticate the connection. This failure is to prevent a class of security attacks.

3.1.2.2.2 Use of Authentication

The following section describes how authentication works in Windows.

3.1.2.2.2.1 General Model

Windows generally follows the GSS model of security. In the GSS model, application protocols try to be as agnostic as possible for the specific forms of authentication used in the session. In return, the security protocol (or mechanism) limits its specific behavior to some well-defined interfaces and communicates in opaque messages, which are referred to as *tokens* in the GSS documents.

Therefore, the application is responsible for calling the security runtime to secure its protocol. Conversely, the application does not need to be aware of any specifics of the security protocol that it has selected. This behavior leads to the somewhat surprising condition that an application protocol may not know which security protocol it is using in some conditions.

3.1.2.2.2.2 Known Idiosyncrasies

The following protocols have specific idiosyncratic behavior with regard to use of authentication protocols. In most cases, this is due to externally mandated behavior or certain legacy constraints.

3.1.2.2.2.2.1 Server Message Block (SMB) and Common Internet File System (CIFS)

Server Message Block (SMB) is the oldest general application protocol in Windows, and it predates the movement toward the GSS authentication model and has the NTLM built into its session establishment messages. SMB uses NTLM when the server is not compatible with Windows 2000 or later, or when SPNEGO has determined that the server is not expected to use a protocol other than NTLM. In this case, the SMB client extracts the response values from the NTLM response token and directly supplies these in the message that is used to authenticate to the SMB server.

SMB uses a Windows extension to SPNEGO in which the server can create a hint to the client. The hint is not critical to the success of the protocol. The hint can help the client determine the best security protocol to use with the server. The hint allows the client to determine the following:

- The set of mechanisms available on the server.
- The claimed domain of the server.

3.1.2.3 Authorization

After an identity is suitably authenticated, the natural next step is to use that identity to authorize access to a resource. Windows has a very expressive authorization model available for applications and system components to use for making authorization decisions.

Windows was originally designed to meet the requirements of the C2 level of the Trusted Computer System Evaluation Criteria (TCSEC). The TCSEC program has since been supplanted by profiles written under the Common Criteria for Information Technology Security Evaluation, such as the Controlled Access Protection Profile.

The C2 requirements (and later the CAPP requirements) for authorization are centered on *discretionary* access control. For discretionary access control, the owner of a particular resource (or a delegate of the owner) determines what access others should have. This is in contrast to *mandatory* access control schemes in which another party maintains control over the resource regardless of the expectations of the owner.

3.1.2.3.1 Resource Managers

Windows meets these requirements for discretionary access control by providing a single access evaluation routine that any number of *resource managers* can invoke. Many resource managers—including the file system, registry, Active Directory, and operating system constructs such as processes—exist in a Windows-based system. Even though these resource managers control very different objects, they share a common method for controlling access.

In the Windows authorization model, a resource manager is the code or component that implements one or more object types. The NTFS file system is a resource manager that implements files and directories; the Windows registry is a resource manager that implements keys. To participate in the authorization scheme, the resource manager is required to maintain a *security descriptor* with each object that is protected. The resource manager merely needs to maintain the storage for the security descriptor and is not required to understand the contents.

The Windows Security Overview also distinguishes between ordinary objects in the resource manager and *containers* exposed by the resource manager. In the file system example, files are objects, and directories are containers. This distinction is important during the creation of new objects.

3.1.2.3.2 Security Descriptors

The security descriptor is a collection of four main elements. The *owner* field is a SID that specifies the owner of the resource. The *group* field specifies the group associated with the resource. The group field is not evaluated by Windows components, and it exists for POSIX compatibility. The *DACL* field specifies the discretionary access control list, and the *SACL* field specifies the system access control list (SACL).

When associated with a resource, the security descriptor is intended to be opaque. The resource manager should never be required to examine the contents of the security descriptor. The security descriptor fields can be used by the resource manager for other purposes, however. For example, the file system can implement a storage quota system by using the owner field associate resources consumed with an owner for billing.

3.1.2.3.3 Discretionary Access Control Lists (DACLs) and Access Control Entries (ACEs)

Discretionary access control lists (DACLs, but often shortened to ACLs) form the primary means by which authorization is determined. An ACL is conceptually a list of <*account, access-rights*> pairs, although they are significantly richer than that.

Each pair in the ACL is termed an access control entry (ACE). Each ACE has additional modifiers that are primarily for use during inheritance. There are also several different kinds of ACEs for representing both access to a single object (such as a file) and access to an object with multiple properties (such as an object in Active Directory).

The ACE contains the SID of the account to which the ACE pertains. The SID can be for a user or a group.

Windows supports both positive ACEs, which grant or allow access rights to a particular account, and negative ACEs, which deny access rights to a particular account. This allows a resource owner to specify, for example, *grant read-access to group Y, except for user Z.*

DACLs can be configured at the discretion of any account that possesses the appropriate permissions to modify the configuration, including Take Ownership, Change Permissions, or Full Control permissions. DACLs consist of the following elements.

Header: Metadata pertaining to the ACEs associated with the DACL.

SID (user): The SID of the owner of the object.

SID (group): The SID of the built-in Administrators or Domain Admins group if the account that owns the object is a member of either of these groups.

Access control entry (ACE): A DACL can contain several ACEs. The following list identifies and describes these terms:

- **Explicit allow ACE:** An ACE applied directly to the resource that grants access. An explicit allow will always override an inherited deny but will always be overridden by explicit deny ACEs.
- **Explicit deny ACE:** An ACE applied directly to the resource that denies access. An explicit deny will always override all other permissions.
- **Generic deny ACE:** An ACE that denies access to an account or security group based on the user or group SID. A generic deny ACE can be inherited from the object's parent or assigned directly to

the object. The generic deny ACE is specific to the object and child objects of the same class, based on the security settings defined in the object class in the schema.

Generic allow ACE: An ACE that allows access to an account or security group based on the user or group SID. A generic allow ACEs can be inherited from the object's parent or assigned directly to the object. The generic allow ACE is specific to the object and child objects of the same class, based on the security settings defined in the object class in the schema.

Inherited deny ACE: An ACE inherited from the resource's parent object. An inherited deny ACE overrides an inherited allow permission, but is overridden by an explicit allow.

Inherited allow ACE: An ACE inherited from the resource's parent object.

Object-specific deny ACE: An ACE used within Active Directory that does one of the following:

- denies access to a property on the Active Directory object
- denies access to a property set on the Active Directory object
- limits the ACE inheritance to a specified type of child object based on the SID(s) of the child or children.

The object-specific deny ACE can be inherited from the object's parent or assigned directly to the object. Also, the object-specific deny ACE applies to specific classes of child objects.

Object-specific allow ACEs: An ACE used within Active Directory that does one of the following:

- Allows access to a property on the Active Directory object
- Allows access to a property set on the Active Directory object
- limit the ACE inheritance to a specified type of child object based on SID(s) of the child or children.

The object-specific allow ACE can be inherited from the object's parent, or assigned directly to the object/ Also, the object-specific allow ACE applies to specific classes of child objects.

When access is requested to an Active Directory object, the Local Security Authority (LSA) compares the access token of the account that is requesting access to the object to the DACL. The security subsystem checks the object's DACL, looking for ACEs that apply to the user and group SIDs referenced in the user's access token. The security subsystem then steps through the DACL until it finds any ACEs that allow or deny access to the user or to one of the user's groups. The subsystem does this by first examining ACEs that have been explicitly assigned to the object and then examining ones that have been inherited by the object. The following illustration shows the important parts of an access token and a DACL when a request is evaluated.

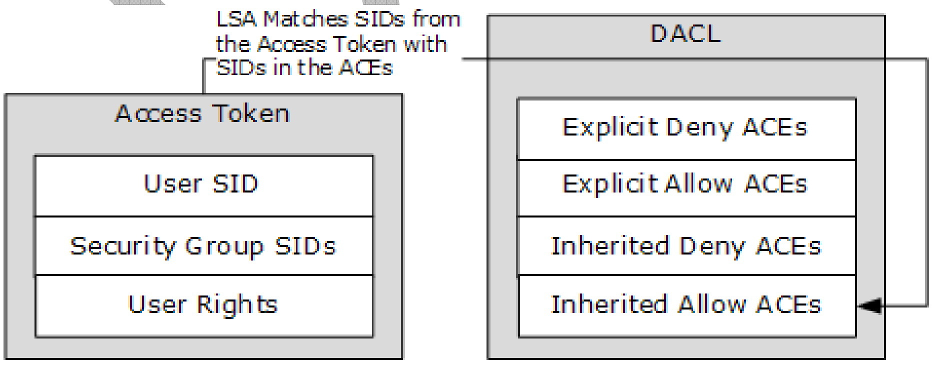


Figure 9: Evaluation process for access tokens against a DACL

If an explicit deny is found, access is denied. Explicit deny ACEs are always applied, even if conflicting allow ACEs exist. Explicit allow ACEs are examined, as are inherited deny and allow ACEs. The ACEs that apply to the user are accumulated. Inherited deny ACEs overrule inherited allow ACEs but are overruled themselves by explicit allow permissions. If none of the user SIDS or group SIDs in the access token match the DACL, the user is denied access implicitly.

In Windows, a security principal's level of access to files and folders is determined by NTFS file system and share permissions. These permissions are discretionary; anyone with ownership of a file or folder, Change permissions, or Full Control permissions can assign access control at their discretion. When newly installed, Windows assigns default permission structures to operating system files and folders, but a user might need to alter these permissions to meet specific security requirements.

When a user attempts to access a file or folder on an NTFS partition, the user's access token is compared with the DACL of the file or folder. If no ACEs correspond to a SID in the user's access token, the user is implicitly denied access to the resource. If ACEs correspond to the user's access token, the ACEs are applied in the following order:

1. Explicit deny
2. Explicit allow
3. Inherited deny
4. Inherited allow

ACEs that apply to the user are cumulative, meaning that the user will receive the sum of the ACEs that apply to his or her user account and groups of which the user is a member. For example, if an ACL contains two allow ACEs that apply to the user, one for Read access and the other for Write access, the user will receive Read and Write access.

3.1.2.3.4 Access Rights

Different resource managers and resource types have different access rights. Files may have read and write access, but processes have entirely different rights such as terminate. However, all resource managers use the same formats for encoding access rights in the ACEs. This is done by allowing the resource managers to define their own specific access rights.

Windows accomplishes this by partitioning the access rights space. All access rights are encoded into a single, 32-bit value in the ACE. The most significant 16 bits are considered standard access rights and are common across all resource managers. These rights include Delete access, Generic-Read access, and other similar rights. These rights are either expected of all resource managers (such as Delete) or are used in a way that allows programs to work with multiple resource managers in a similar manner.

The least significant 16 bits are termed object-specific and are meaningful only to the resource manager that defines them. Thus the file system may define that bit 1 indicates the capability to read the file and that bit 2 indicates the capability to write the file, whereas the registry may define bit 1 to be enumerate subkeys and bit 2 to be read a key's value.

3.1.2.3.5 Authorization

Windows has a single method in the system for determining access. In that way, the results are always predictable and consistent. The process is as follows.

To determine access, the calling resource manager supplies the security descriptor (which contains the ACL) with the identity of the user and all the groups of which the user is a member, and the access requested by the user. For this example, the following values are used.

```
Security Descriptor: Owner: U1, DACL: <<U2, Read>, <G1, Read>,
 <G2, Write>>
Identity: <U1, G2>
Access Request: Write
```

In this example, the security descriptor has an ACL that grants U2 Read access, G1 Read access, and G2 Write access. The identity of the user making the request is U1, and that user is a member of the group G2 as well. The request is for Write access.

When processing this request, Windows iterates through the entries in the ACL, testing against the identity. If the identity in the ACE matches one of the identities of the user, the ACE is examined further. In this example, the first two ACEs do not match any identity, and so they are skipped. The third ACE applies (G2 matches), and then the granted access rights are compared against the access request. They match, and the user is therefore granted access.

As noted earlier, multiple access rights are encoded together, so the access request could be for both Read access and Write access. In the preceding example, access would be denied because G2 was granted only Write access.

All the requested rights do not have to be granted by a single ACE. Consider the following example.

```
Security Descriptor: Owner:U1, DACL:<<U2,Read>,<G1,Read>,<G2,Write>>
Identity:<U1,G1,G2>
Access Request: Read,Write
```

The processing would be as follows:

> The first ACE does not match, and so it is skipped. The second ACE now does match and is therefore examined further. The granted access is removed from the access request, in this case, Read. There are still values left in the access request, so processing continues. The third ACE matches (on G2) and grants Write access. The granted access, Write, is removed from the access request, but now there are no remaining access requests. The access is granted, and processing stops.

3.1.2.3.6 Inheritance

The Windows authorization model supports a concept of inheritance by which new objects can inherit one or more ACEs from their parent container. In practice, this allows an administrator to establish default security on, for example, a directory, and all new files that are created in that directory receive a preset ACL. Although the owner of the file can still override that ACL and establish its own, if nothing is done (through the premise of discretionary access control), the default is as the administrator wants.

One attribute that can be applied to ACEs is the *Object-Inherit* flag. This flag indicates that when a new object is created, this ACE should be carried forward to the security descriptor of the new object. An additional flag, *Container-Inherit*, indicates that new containers created under this container should receive this ACE. For the file system, this allows different default ACLs for directories as opposed to files.

3.1.2.3.7 System Access Control Lists

A system access control list (SACL) enables administrators to log attempts to access a secured object. Like a DACL, an SACL is a list of ACEs. Each ACE specifies the types of access attempts by a specified account that cause the system to generate a record in the security event log. An ACE in an SACL can generate audit records when an access attempt fails, when it succeeds, or both.

3.1.2.3.8 Windows Integrity Mechanism

The Windows integrity mechanism extends the security architecture by defining a new access control entry (ACE) type to represent an integrity level in an object's security descriptor. The new ACE represents the object integrity level. An integrity level is also assigned to the security access token when the access token is initialized. The integrity level in the access token represents a subject integrity level. The integrity level in the access token is compared against the integrity level in the security descriptor when the security reference monitor performs an access check. Windows Vista operating system uses the AccessCheck function to determine what access rights are allowed to a securable object. Windows restricts the allowed access rights depending on whether the subject's integrity level is higher or lower than the object, and depending on the integrity policy flags in the new access control ACE. The security subsystem implements the integrity level as a mandatory label to distinguish it from the discretionary access under user control that ACLs provide.

Note: The mandatory integrity labels are not evaluated as part of the descretionary access control by the AccessCheck algorithm (see [MS-DTYP] Section 2.5.2.1). These SIDs are not set in the token with the SE_GROUP_ENABLED attributes, which means that even though these SIDs are added to the SIDS_AND_ATTRIBUTES array in the token, they are not evaluated as part of sidintoken.

3.1.2.4 Impersonation

In distributed systems, it is typical for one server to call another server to accomplish a task for a client. This functionality is called **impersonation**. To handle these requests for a client, the server must be given the authority to do so. The ability to call other servers while impersonating the original client is called **delegation**.

Through impersonation, a thread runs in a security context that is different from the context of the process that owns the thread. When a server thread runs in the security context of the client, it uses an access token that represents the client credentials in order to obtain access to the objects to which the client has access. This provides the ability for a thread to run by using different security information from the process that owns the thread. Typically, a thread in a server application impersonates a client. This impersonation allows the server thread to act for that client in order to access objects on the server or validate access to the client objects.

The following diagram shows the impersonation process. A client makes a request to server A. If server A must query server B to complete the request, server A impersonates the client security context and makes the request to server B for the client. Server B uses the security context of the original client, instead of the security identity for server A, to determine whether to complete the task.

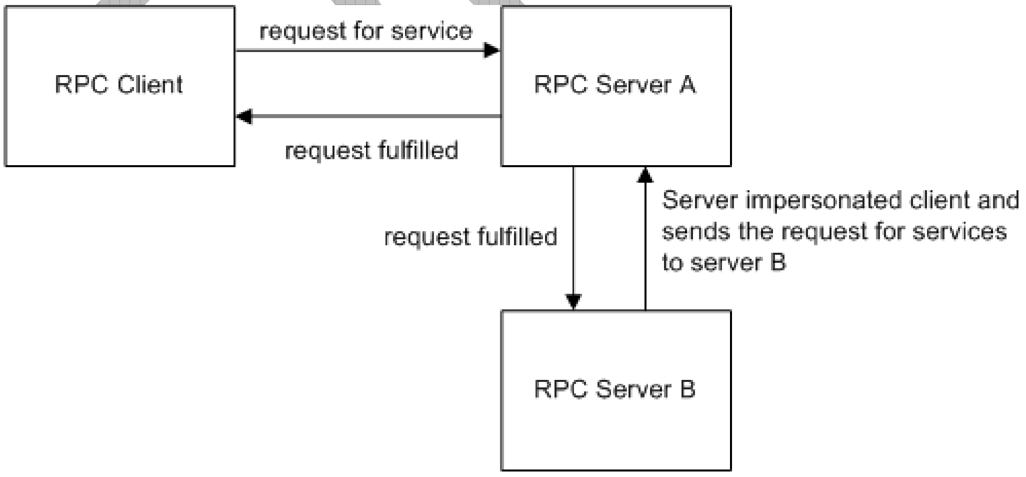


Figure 10: Impersonation process

When delegation is used, a server that is impersonating a client can call another server and can make network calls by using the credentials of the client. From the perspective of the second server, requests that come from the first server are indistinguishable from requests that come from the client. Not all security providers support delegation. Windows provides only one security provider that supports delegation: the Kerberos protocol.

Delegation must be implemented with caution due to the privileges that the client gives the server during an RPC. To address this, the Kerberos protocol allows calls that use the impersonation level of delegation only if mutual authentication is requested. Domain administrators can limit the computers to which calls with delegation impersonation level are made to prevent unsuspecting clients from making calls to servers that can abuse their credentials.

Calls that use **ncalrpc** [MSDN-NCALRPC] are an exception to the delegation rule. When these calls are made, the server gets delegation rights, even if an impersonation level of impersonate is specified. That is, a server can call other servers on behalf of the client. This works for one remote call only. For example, if client A calls local server LB using **ncalrpc**, local server LB can impersonate and call remote server RB. Remote server RB can act for client A, but only on the remote computer on which remote server RB is running. Local server LB cannot make another network call to remote computer C unless LB specifies an impersonation level of delegate when it calls RB.

A primary use of impersonation is to perform access checks against the client identity. Using the client identity for access checks can cause access to be either restricted or expanded, depending on what the client has permission to do. For example, a file server might have files that contain confidential information and each of these files is protected by an ACL. To help prevent a client from obtaining unauthorized access to information in these files, the server can impersonate the client before accessing the files.

3.1.2.4.1 Cloaking

There are two important security considerations regarding impersonation and delegation:

- What should the server be allowed to do when acting for the client?
- What identity is presented by the server when it calls other servers for a client?

In Windows, the Component Object Model (COM) provides the functionality that is explained here. The client can set an impersonation level that determines the extent to which the server can act as the client. If the client grants enough authority to the server, the server can impersonate (pretend to be) the client. When the server impersonates the client, it is given access to only those objects or resources that the client has permission to use. The server, acting as a client, can also enable cloaking in order to mask its own identity and project the client identity in calls to other COM components.

The following figure illustrates impersonation without and with cloaking. A and B represent two processes running on machine 1, and C represents a process that is running on machine 2. Process A calls B, and B calls C. Client A sets the impersonation level. B sets the cloaking capability. If A sets an impersonation level that permits impersonation, B can impersonate A when calling C on A's behalf. The identity that is presented to process C is either A's identity or B's identity, depending on whether cloaking was enabled by B. If cloaking is enabled, the identity that is presented to process C is A. If cloaking is not enabled, B's identity is presented to C.

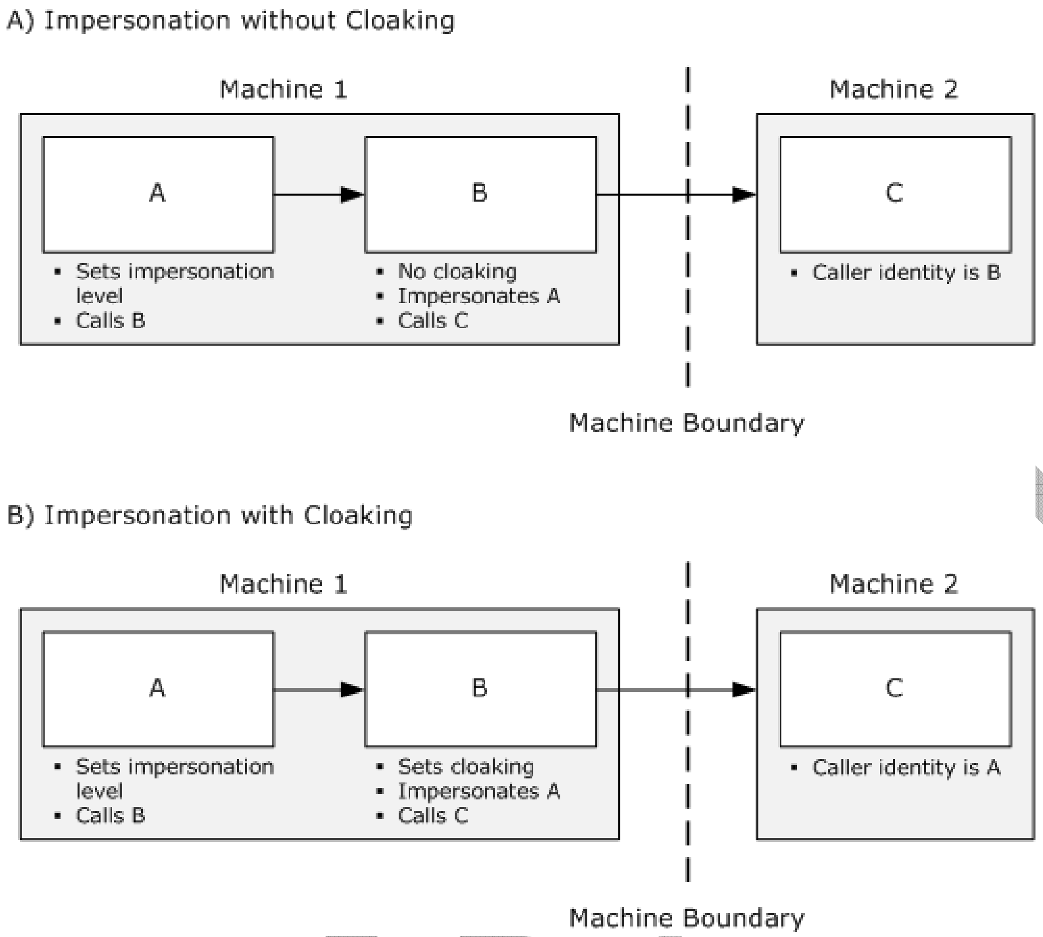


Figure 11: Impersonation without and with cloaking

3.1.2.4.2 Impersonation Tokens

Access tokens are objects that describe the security context of a process or a thread. They provide information that includes the identity of a user account and a subset of the privileges that are available to the user account. Every process has a *primary access token* that describes the security context of the user account that is associated with the process.

By default, the system uses the primary token when a thread of the process interacts with a securable object. However, when a thread impersonates a client, the impersonating thread has both a primary access token and an impersonation token. The impersonation token represents the security context of the client, and this access token is the one that is used for access checks during impersonation. When impersonation is complete, the thread reverts to using only the primary access token.

3.1.2.4.3 Impersonation Levels

Windows provides various degrees of impersonation through impersonation levels, which indicate how much authority is given to the server when it is impersonating the client.

Currently, there are four impersonation levels are available: anonymous, identify, impersonate, and delegate. <13> The following list briefly identifies and describes each impersonation level:

- **Anonymous level (RPC_C_IMP_LEVEL_ANONYMOUS)**: The client is anonymous to the server. The server process can impersonate the client, but the impersonation token does not contain any information about the client. This level is supported only over the local interprocess communication transport. All other transports silently promote this level to identify.
- **Identify level (RPC_C_IMP_LEVEL_IDENTIFY)**: The system default level. The server can obtain the identity of the client, and the server can impersonate the client in order to do ACL checks.
- **Impersonate level (RPC_C_IMP_LEVEL_IMPERSONATE)**: The server can impersonate the security context of the client while acting for the client. The server can access local resources as the client. If the server is local, it can access network resources as the client. If the server is remote, it can access only resources that are on the same machine as the server.
- **Delegate level (RPC_C_IMP_LEVEL_DELEGATE)**: The most powerful impersonation level. When this level is selected, the server (whether local or remote) can impersonate the security context of the client while acting on behalf of the client. During impersonation, the client credentials (both local and network) can be passed to any number of machines.<14> This level is supported beginning with Windows 2000

  For impersonation to work at the delegate level, the following requirements must be met:

  - The client must set the impersonation level to RPC_C_IMP_LEVEL_DELEGATE.
  - The client account must not be marked "Account is sensitive and cannot be delegated" in Active Directory.
  - The server account must be marked with the "Trusted for delegation" attribute in Active Directory.
  - The computers that host the client, the server, and any "downstream" servers must all be running Windows 2000 in a Windows 2000 domain.

By choosing the impersonation level, the client tells the server how far it can go in impersonating the client. The client sets the impersonation level on the proxy that it uses to communicate with the server.

3.1.2.4.4 Setting the Impersonation Level

There are two ways to set the impersonation level:

- The client can set the impersonation level processwide through a call to CoInitializeSecurity.
- A client can set proxy-level security on an interface of a remote object through a call to IClientSecurity::SetBlanket (or the helper function CoSetProxyBlanket).

The impersonation level is set by passing an appropriate RPC_C_IMP_LEVEL_xxx value to CoInitializeSecurity or CoSetProxyBlanket through the *dwImpLevel* parameter.

Different authentication services support delegate-level impersonation to a different extent. For example, NTLMSSP in Microsoft Windows® 2000 supports cross-thread and cross-process delegate-level impersonation but not cross-machine. However, the Kerberos protocol (implemented by Windows 2000) supports delegate-level impersonation across machine boundaries, whereas SChannel does not support any impersonation at the delegate level. If there is a proxy at the impersonate level and the implementer wants to set the impersonation level to delegate, IClientSecurity::SetBlanket SHOULD be called by using the default constants for every parameter except the impersonation level. COM chooses NTLM locally and the Kerberos protocol remotely (when the Kerberos protocol is supported).

3.1.2.4.5 Windows API Impersonation Functions

The Windows API provides the following functions to begin an impersonation:

- A Dynamic Data Exchange (DDE) server application can call the DdeImpersonateClient function to impersonate a client.
- A named pipe server can call the ImpersonateNamedPipeClient function.
- The ImpersonateLoggedOnUser function can be called to impersonate the security context of an access token for a logged-on user.
- The ImpersonateSelf function enables a thread to generate a copy of its own access token. This is useful when an application needs to change the security context of a single thread. For example, sometimes only one thread of a process needs to enable a privilege.
- The SetThreadToken function can be called to cause the target thread to run in the security context of a specified impersonation token.
- An RPC server application can call the RpcImpersonateClient function to impersonate a client.
- A security package or application server can call the ImpersonateSecurityContext function to impersonate a client.

For most of these impersonations, the impersonating thread can revert to its own security context by calling the RevertToSelf function. The exception is the RPC impersonation, in which the RPC server application calls RpcRevertToSelf or RpcRevertToSelfEx to revert to its own security context.

For more information about the Windows API impersonation functions, visit the MSDN home page.

3.1.3 File, Print, and Fax Concepts

Fax Services, Print Services, and File Access Services all require foundational knowledge of:

- SMTP: The Simple Mail Transfer Protocol, as defined in [RFC821]
- TIFF: Tagged Image File Format, as defined in [RFC3302]
- EMFPLUS: Enhanced Metafile Format Plus Extensions, as defined in [MS-EMFPLUS]
- Object Stores (NTFS, FAT, FAT32)
- Accessing a File Share through NFS Access Protocols or SMB Access Protocols

The following concepts are important to understanding the usage of the File Access System and the Print Services System, as described in [MS-FSSO] and [MS-PSSO], respectively:

- Group Policy
- Active Directory
- Windows® BranchCache™

The following section compares and contrasts concepts that are common to the File, Print, and Fax protocols.

3.1.3.1 File Access Services and Print Services Commonalities

The Print Services System and File Access Services are related as a result of a mutual dependency: the Print Services System uses the SMB Services component within the File Access Services System, and the SMB Services component uses the Print Services System.

The Print Services System uses the SMB Services component within the File Access Services System for the following tasks:

- When the Print Client copies printer driver files and printer driver packages from Print Server by calling local APIs of the SMB Services component, which remotely uses the SMB protocol family.
- In certain configurations and scenarios, when the Print Client sends print jobs to Print Server by calling local APIs of the SMB Services component, which remotely uses the SMB protocol family.
- In certain configurations and scenarios, when the Print Client monitors print queue and job status on the Print Server by calling local APIs of the SMB Services component, which remotely uses the protocol described in [MS-RAP].

The SMB Services component within the File Access Services System uses the Print Services System for the following tasks:

- To route print jobs received via the SMB protocol family to a local API of the Print Spooler running on the same server.
- To route print queue and job status monitoring requests received via the protocol described in [MS-RAP] to a local API of the Print Spooler running on the same server.

These mutual dependencies require that the subsystems supporting the Print Services System and File Access Services are installed together on clients and servers.

3.1.3.2 Network Printing and Internet Printing

The Print Services System provides support for a Print Client sending print jobs to a printer using a private network as well as to a printer using the internet. Network printing and internet printing use different protocols providing different capabilities.

Network printing: A Print Client within a private network can send a print job to a printer attached to a Print Server within the same network. Printing within a private network includes the authentication of a Print Client's right to use the Print Server, supporting a level of security that allows print clients to use RPC-based printing protocols to communicate with a Print Server, such as RPRN (RPC/named SMB pipes), PAR (RPC/TCP), and PAN (RPC/TCP).

Internet printing: A Print Client can send a print job across the internet to a printer attached to a Print Server connected to the internet. The Print Server may be within a private network connected to the internet by a server providing a firewall. Printing across the internet typically involves sending a print job or a print driver file across a firewall that prohibits the use of RPC-based protocols for security reasons. Consequently, the Print Services System supports alternate protocols based on HTTP/HTTPS for sending a print job to the print server (standard IPP), and for downloading a printer driver file from the print server (WPRN). The Print Services System also supports internet printing using the standard LPD/LPR protocol.

3.1.3.3 Fax Services and Print Services Distinctions

The protocol described in [MS-FAX] enables communication between a client system and a FAX Server. Use of the FAX protocol is completely distinct from use of the printing-related protocols described in the

Print Services System documented in [MS-PSSO], though both are RPC-based, as are many other Windows protocols. Although a single shared fax driver appears locally in the user interface on a client system (enumerated as a 'printer' driver), no [MS-PSSO] protocols are used to interact with that driver or to support any FAX use cases. The fax printer driver uses only the [MS-FAX] protocol to transfer faxes from the client system to a FAX Server, without relying on any member protocol of the Print Services System.

The primary component described by the Print Services System is a print spooler, responsible for receiving the print output of an application, buffering it, and sending the print data to a shared print queue managed by the print spooler. Print queues are associated with printer drivers that support the creation of application print output and the translation of metafiles, allowing applications to obtain metrics and status information about printers. The Windows implementations of the Print Client and Print Server roles are provided by the print spooler component. Each Windows system runs a print spooler. Therefore, each Windows computer can act as a Print Client or a Print Server.

The primary components described by the Fax Services are client-based components for composing fax documents, including selecting and editing fax cover pages, and server-based components for queuing, routing, transmitting, and archiving fax documents, as well as a fax console used to configure the Fax Server using protocol methods described in [MS-FAX]. The fax functionality embodied in the shared fax driver on a client wrap the fax job in an RPC-based wrapper and then forward the job, using RPC interfaces, to the RPC Interface Module of the Fax Server, and then to the outgoing fax queue.

The fax queues on a Fax Server described in [MS-FAX] provide different capabilities than the print queues described in [MS-PSSO], though all are called “queues.” A Fax Server has a single outgoing queue of faxes awaiting transmission. The Fax Server’s outbound routing module dynamically determines, based on rules installed by default or configured from the fax console, which fax device or device group to use to transmit each fax job stored in the single outgoing queue. A Print Server described in [MS-PSSO] has multiple print queues associated with multiple devices. The client of the Print Server chooses a single print queue to be used for any particular print job.

A Fax Server also has a single incoming queue holding received faxes that have not yet been delivered to their final destinations. The Fax Server uses inbound fax routing extensions to process received faxes and deliver them to their destinations. Of three standard fax routing extensions, one extension delivers a fax to a printer, and does so by making local API calls to the Print Services.

Fax queues use only the .TIFF data format to store sent and received fax jobs, though fax queues also use a variety of queue data formats (.FQP, .FQE, .FQR, .FQT) defined by Fax Services to support routing and transmission of fax jobs. To print an inbound fax, the Fax Services use local print spooler API calls and printer drivers to send the fax to a print queue in a manner identical to that of other printing applications. The .TIFF data format is converted locally to a print system specific data format.

3.1.4 Systems Management Concepts

3.1.4.1 Managed Host

The Windows Management System, as discussed in this document, provides management of individual computers, called managed hosts, and also provides management of individual objects on those hosts. The Windows Protocols System provides not only the ability to monitor host functions and health, but also to modify the behavior of certain of those functions through the Windows Management System and the Group Policy System.

3.1.4.2 Managed Object

A managed object is a hardware or software component of a managed host. The Windows Protocols System provides not only the ability to monitor the behavior and history of an object, but also to modify certain behaviors of the object through the Windows Management System and the Group Policy System.

3.1.4.3 Group Policy

Group Policy is an infrastructure used to deliver and apply one or more desired configurations or policy settings to a set of targeted users and computers within an Active Directory environment. This infrastructure consists of a Group Policy engine and multiple client-side extensions (CSEs) responsible for writing specific policy settings on target client computers.

Group Policy settings are contained in Group Policy objects (GPOs), which live in the domain and can be linked to Active Directory sites, domains, or organizational units (OUs) containers. The settings within GPOs are then evaluated by the affected targets through Active Directory.

Group Policy is one of a group of systems management technologies that provides users with consistent access to their applications, application settings, roaming user profiles, and user data, from any managed computer — even when they are disconnected from the network.

Core Group Policy or the Group Policy engine is the framework that handles common functionalities across Administrative Template settings and other client-side extensions.

3.1.5 Application Services Concepts

For implementations that use the Transaction Processing System and/or the Message Queueing System, the following background knowledge as described in [GRAY] is required:

- ACID transactions
- Transaction processing concepts
- Concept of transaction managers, applications, and resource managers in transaction processing
- Two-Phase Commit protocol
- Transaction trees, root transaction manager, superior and subordinate participants
- Transaction marshalling, transaction pull and push propagation, and transaction recovery

Additionally, the understanding of the following concepts is required, as described in [MS-TPSO] sections 3.1.3, 3.1.4, and 3.1.5:

- Phase Zero
- Single-Phase Commit
- System Base and System External protocols

3.1.6 Web Services Concepts

Several Windows Protocols Systems protocols, such as those specified in [MS-RMPR], [MS-WUSP], and [MS-WSMAN] are implemented as Web Services protocol using SOAP and Web Services Description Language (WSDL). Other protocols in the Windows Protocols System, such as the one specified in [MS-LWSSP], are extensions or profiles of SOAP and Web Services protocols such as WS-Trust.

For more information on Web Services, see the W3C web site http://www.w3.org.

The following table lists the documents describing Microsoft's Web Services protocols, and their associated System Documents where applicable.

Document Short Name	System Document	Protocol Name
[MS-ADCAP]	None	Active Directory Web Services: Custom Action Protocol Specification
[MS-ADDM]	None	Active Directory Web Services: Data Model and Common Elements
[MS-BPDP]	None	Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Discovery Protocol Specification
[MS-CTAP]	None	CardSpace Token Acquisition Protocol
[MS-DPWSSN]	None	Devices Profile for Web Services (DPWS): Size Negotiation Extension
[MS-DSML]	None	Directory Services Markup Language (DSML) 2.0 Protocol Extensions
[MS-LWSSP]	None	Lightweight Web Services Security Profile
[MS-MWBE]	None	Microsoft Web Browser Federated Sign-on Protocol Extensions
[MS-MWBF]	None	Microsoft Web Browser Federated Sign-on Protocol Specification
[MS-WSDS]	[MS-ADSO]	WS-Enumeration: Directory Services Protocol Extensions
[MS-WSMAN]	[MS-WMSO]	Web Services Management Protocol Extensions for Windows Server 2003
[MS-WSMV]	[MS-WMSO]	Web Services Management Protocol Extensions for Windows Vista
[MS-WSPELD]	[MS-ADSO]	WS-Transfer and WS-Enumeration Protocol Extension for Lightweight Directory Access Protocol v3 Controls Specification
[MS-WSTEP]	[MS-CAESO]	WS-Trust X.509v3 Token Enrollment Extensions
[MS-WSTIM]	None	WS-Transfer: Identity Management Operations for Directory Access Extensions
[MS-WSRVCAT]	[MS-TPSO]	WS-AtomicTransaction (WS-AT) Version 1.0 Protocol Extensions
[MS-XOPP]	None	XML-binary Optimized Packaging (XOP) Profile

3.2 System Purposes

General purpose computer operating systems can provide many functions. The Windows Protocols System is oriented toward computers acting as servers to Windows clients and other compatible server or client software using Communications Protocols, which means the set of rules for information exchange to accomplish predefined tasks between a Windows operating system product and a server operating system product connected via a network.

Microsoft Communications Protocol-based systems provide services based on Microsoft proprietary protocols (including extensions to industry-standard or other published protocols) and in conjunction with publicly available standards based protocols such as TCP/IP, HTTP, HTTPS, SMTP, and FTP. These systems are not oriented toward server to server protocols, nor client to device protocols. These do not include any printer protocols or printer-specific commands or content. They are also not oriented towards communication solely within a single computer not using a network.

Systems using Microsoft Communications Protocols provide server software that interoperates with Windows desktop operating systems. The mission is to provide the following services to clients:

- Authentication Services – provide account authentication, authorization, access control, policy enforcement, data-packets usage accounting, and data-packet auditing services between a client and a server.
- Collaboration Services – provide audio and/or video conferencing, instant messaging, white boarding, application sharing, and similar services.
- File Services - communicate data packets from data stored on storage media on other network-attached computers.
- Network Access Protection Services **-** communicate client health status information, digital certificate requests, and digital certificate issuance or refusal data packets between a client and server to accomplish system health evaluation goals and aid in ensuring the health and protection of networked systems.
- Multiplayer Games Services - transfer multiplayer game data is over the network.
- Proxy/Firewall/NAT Services - examine and reject or forward data packets based on predefined permissions.
- Systems Management Services - provide remote systems management functionality.
- Virtual Private Network Services **-** provide functionality for communicating between a VPN server and a VPN client, including the following data communication:
  - Authentication, policy, authorization, and connection data packets as part of a private network connection.
  - Secure private network data packets encoded for IPSec network address translator traversal.
- Windows Update Services – communicate software update information, software update metadata, and XPRESS compression data between a Windows Server Update Services (WSUS) implementation and a Windows Client configured to communicate with the Windows Update Services (WUS).
- Certificate Services - provide X.509 certificate requests, renewal, key archival, certificate issuance, and lifecycle-management services.
- Digital Rights Management Services – access and generate license requests relating to digital rights management.
- General Services include the following:
  - Transaction Processing Services used to communicate with a transaction manager to perform distributed transaction coordination, driving the transaction life cycle.
  - Message Queuing Services that enable reliable and secure **asynchronous messaging** between applications over a variety of deployment topologies.
  - Networking and transports provide a general-purpose set of protocols that are not mapped to a specific service.

- Media Streaming Services – for communicating data packets that originate from downloadable and streaming audio, visual, and other multimedia data files.
- Print and Fax Services - manage the interaction between file/print and fax servers and clients including authentication to the print/fax server, notification of print/fax status, and management of remote print jobs.
- Rights Management Services - provide support for information protection through content encryption and fine-grained policy definition and enforcement thus enabling end users to create and consume protected information.
- Terminal Services - communicate remote graphical desktop interaction and display data packets, and sound, file redirection, and print redirection data packets from applications accessed by authorized clients to a Terminal Server.
- Web Services - manage and communicate with servers that provide Internet services to:
  - Communicate with remote services.
  - Provide remote services that can be accessed by existing clients.
  - Provide management interfaces (for both content and administrative data) for the documented services.

3.3 System Use Cases

This section lists the set of use cases that span the functionality of the Windows Protocols System. The following tables summarize the use cases described in the System Documents.

3.3.1 Networking and Transport Protocols Use Cases

[MS-NAPSO]: Network Access Protection System Overview

Use case	Description
Create SoH – Client User (Triggered by Client User)	The goal of this use case is to create the SoH [MS-SOH] when the Client User triggers a new SoH transaction.
Create SoH – Client User (Triggered by PEP)	The goal of this use case is to create the SoH for the Client User when PEP triggers a new SoH transaction.
Create SoH – Client User (Triggered by Group Policy)	The goal of this use case is to create the SoH for the Client User when a change of the Group Policy on the computer triggers a new SoH transaction.
Create SoH – Client User (Triggered by Network Status)	The goal of this use case is to create the SoH for the Client User when a change in the network connectivity status of the Client Computer triggers a new SoH transaction.
Send SoH via HTTP/S Channel – Client User	The goal of this use case is to send the SoH [MS-SOH] for the Client User via an HTTP/S Channel.
Send SoH via PEAP Channel – Client User	The goal of this use case is to send the SoH [MS-SOH] for the Client User via a PEAP Channel.
Send SoH via DHCP Channel – Client User	The goal of this use case is to send the SoH [MS-SOH] for the Client User via a DHCP Channel.

Use case	Description
Receive State of Health via RADIUS Channel	The goal of this use case is to receive the SoH [MS-SOH] on the NAP health policy server via RADIUS Channel.
Evaluate Health – PEP	The goal of this use case is to evaluate health using the received SoH.
Create SoHR – PEP	The goal of this use case is to create a SoHR after the evaluation of health on the PDP.
Send SoHR via RADIUS Channel – PDP	The goal of this use case is to send the SoHR [MS-SOH] and enforcement decision from the PDP to PEP via a RADIUS Channel.
Restrict Network Access of the NAP Client Using the DHCP Channel – PDP	The goal of this use case is to restrict the network access of a NAP Client using DHCP enforcement or DHCP PEP based on the response from the server.
Restrict Network Access of the NAP Client Using the HTTP/S Channel	The goal of this use case is to restrict network access of the client using the HTTP/S channel or an HTTP/S-based PEP based on the response from the server.
Restrict Network Access of the NAP Client Using the PEAP Channel	The goal of this use case is to restrict network access of the client using the PEAP channel or a PEAP-based PEP based on the response from the NAP health policy server.
Provide Full Network Access of the NAP Client Using the DHCP Channel	The goal of this use case is to provide full network access for a client using the DHCP channel or a DHCP-based PEP based on the response from the NAP health policy server.
Provide Full Network Access of the NAP Client Using the HTTP/S Channel	The goal of this use case is to provide full network access for a client using the HTTP/S channel or a HTTP/S-based PEP based on the response from the server.
Provide Full Network Access of the NAP Client Using the PEAP Channel	The goal of this use case is to provide full network access for a client using a PEAP-based PEP based on the response from the NAP health policy server.
Receive SoHR via HTTP/S Channel – PDP	The goal of this use case is to receive SoHR [MS-SOH] messages on the NAP Client via an HTTP/S Channel.
Receive SoHR via PEAP Channel – Client User	The goal of this use case is to receive SoHR [MS-SOH] messages on the NAP Client via a PEAP Channel.
Receive SoHR via DHCP Channel – Client User	The goal of this use case is to receive SoHR [MS-SOH] messages on the NAP Client via a DHCP Channel.
Process SoHR – Client User	The goal of this use case is to process the SoHR [MS-SOH] for the Client User so that the system state of the Client Computer is updated and automatic remediation can be triggered if the Client Computer is unhealthy.
Manual Remediation of the NAP Client – PDP	The goal of this use case is to manually remediate the client based on the response from the server (SoHR). The exact method of remediation depends on the specifics of the SHA/SHV, but this document uses the WSHA/WSHV, as described in [MS-WSH], as an example.
Automatic Remediation of the NAP Client –	The goal of this use case is to automatically remediate the client based on the response from the PDP (the SoHR). The exact method

Use case	Description
PDP	of remediation depends on the specifics of the SHA/SHV, but this document uses the WSHA/WSHV, as described in [MS-WSH], as an example.

3.3.2 Identity and Security Protocols Use Cases

[MS-ADSO]: Active Directory System Overview

Use case	Description
Create Directory Object	This use case represents the creation of a new directory object in the directory tree by a client.
Search for Directory Object	This use case represents the retrieval of information from one or more directory objects from the directory service by the client.
Modify Directory Object	The client alters the values of attributes on a previously created directory object.
Delete Directory Object	This use case represents the deletion of a directory object from the directory service by the client.
Account Creation	This use case represents the creation of a new account in the Active Directory System by a client.
Account Enumeration	This use case represents the enumeration by a client of accounts stored in the Active Directory System.
Account Modification	This use case represents the modification of an existing account stored in the Active Directory System.
Account Removal	This use case represents the removal of an existing account stored in the Active Directory System.
Get Policy	This use case represents the retrieval of a server's security policy settings by a client.
Set Policy	This use case represents the modification of a server's security policy setting by a client.
Name Cracking	This use case represents the translation of a name that identifies a directory object from one format or type of name (such as a distinguished name) to another format or type of name (such as the UPN).
SID Translation	This use case represents the translation of a security identifier (SID) associated with a directory object to a human-readable textual name, and vice versa.

[MS-AUTHSO]: Windows Authentication Services System Overview

Use case	Description
Interactive Domain Logon – User	The user needs to access local and network resources authorized to the user.

Use case	Description
HTTP Access Authentication – Server	The user needs authentication to access protected resources on an HTTP server.
SMB Access Authentication – Server	The user needs to access protected resources on an SMB server.
Remote Procedure Services – RPC Server	The user needs to execute a procedure on an RPC server.

[MS-CAESO]: Certificate Autoenrollment System Overview

Use case	Description
Automatically Enroll and Renew Certificates – Server Administrator	The goal of this use case is to automatically enroll and renew certificates for a computer based on the administrator-defined CEPs.

[MS-CASO]: Certification Authority System Overview

Use case	Description
Enroll for a Certificate – End Entity	The goal of this use case is for the End Entity to be issued a certificate.
Edit CA Configuration Settings – CA Administrator	The goal of this use case is for the CA Administrator to be able to define and edit various configuration settings on the CA that affect behavior and policy around the issuance of certificates.
Revoke a Certificate – CA Administrator	The goal of this use case is to revoke a previously issued certificate and to publish a list of revoked certificates.
Recover Archived Certificate and Key – CA Administrator	The goal of this use case is to recover a certificate and its private key that have been archived within the CA database.

[MS-DISO]: Domain Interactions System Overview

Use case	Description
Locating a Domain Controller – Domain Client	This is the first step when a client is attempting to join a domain or remove itself from the domain. The client must either have the name of the domain controller or know that one is available to be located. The client can be a workstation or a server. On success the name of the domain controller is returned to the client.
Joining a Domain using Predefined Account – Domain Client	This case assumes that the domain controller has been located. The client establishes membership in the domain and then consumes resources that are offered by domain member servers. If the client is also a server, the client publishes necessary information about itself so the other clients in the domain can consume the resources that this client is offering.
Joining a Domain by Creating an Account via SAM – Domain Client	This case assumes that the domain controller has been located. The client establishes membership in the domain and then consumes resources that are offered by domain member servers. If the client is also a server, the client publishes necessary information about itself so the other clients in the domain can consume the resources that this client is offering.

Use case	Description
Joining a Domain by Creating an Account via LDAP – Domain Client	This case assumes that the domain controller has been located. The client establishes membership in the domain and then consumes resources that are offered by domain member servers. If the client is also a server, the client publishes necessary information about itself so the other clients in the domain can consume the resources that this client is offering.
Removing a Domain Member – Domain Client	This case assumes that the domain controller has been located. The client will update its information in the machine object to inform the member servers that it is removing itself. The client then logs off and disconnects from the domain.

3.3.3 File, Print, and Fax Protocols Use Cases

[MS-FSSO]: File Access Services System Overview

Use case	Description
Create Share SMB – Admin Tool	The goal of this use case is to create a share for access via SMB Access Protocols.
Configure Share Directory – Admin Tool	The goal of this use case is to configure file server share directory quota and a file screen.
Create DFS Standalone Namespace – Admin Tool	The goal of this use case is to create a standalone DFS namespace for access via SMB Access Protocols, with [MS-DFSC] extensions.
Create DFS Domain Namespace – Admin Tool	The goal of this use case is to create a domain DFS namespace for access via SMB Access Protocols with [MS-DFSC] extensions.
Create DFS Link – Admin Tool	The goal of this use case is to create a DFS link for access via SMB Access Protocols with [MS-DFSC] extensions.
List Computers – Application	The goal of this use case is to list computers in a workgroup or domain in order to discover computers that may provide file service.
List Shares SMB – Application	The goal of this use case is to list shares on a file server that are accessible via SMB Access Protocols.
List Shares NFS – Application	The goal of this use case is to list shares on a file server that are accessible via NFS Access Protocols.
List Files SMB – Application	The goal of this use case is to list files in a network share directory that are accessible via SMB Access Protocols.
List Files NFS – Application	The goal of this use case is to list files in a network share directory that are accessible via NFS Access Protocols.
Open File SMB – Application	The goal of this use case is to open or create a file in a directory located in an SMB file share, optionally through an [MS-DFSC] mediated namespace.
Open File NFS – Application	The goal of this use case is to open or create a file in an NFS network share directory.

Use case	Description
Perform File Operation SMB – Application	The goal of this use case is to perform a file operation (such as Read, Write or Delete) on a file in an SMB share directory.
Perform File Operation NFS – Application	The goal of this use case is to perform a file operation (such as Read, Write or Delete) on a file in an NFS network share directory.
Act on Directory Change Notification SMB – Application	The goal of this use case is to act on a directory change notification previously requested by the Application.

[MS-PSSO]: Print Services System Overview

Use case	Description
Provision a Print Queue – Administrative Client	The goal of this use case is to make a Print Queue available on a Print Server, subsequently allowing a User to select the Print Queue, establishing all attributes and components necessary to make the Print Queue discoverable and accessible to a print client.
Locate and Establish a Connection to a Print Queue in a Domain Environment – Print Client	The goal of this use case is to make a connection to a shared Print Queue in a Domain environment so documents can be printed by the User.
Locate and Establish a Connection to a Print Queue in a Workgroup Environment – Print Client	The goal of this use case is to make a connection to a shared Print Queue in a workgroup environment so documents can be printed by the User.
Locate and Establish a Connection to a Print Queue from an Internet Client using IPP; Download a Printer Driver Using [MS-WPRN] – Print Client	The goal of this use case is to make a connection to a shared Print Queue from an Internet client so documents can be printed by the User.
Set Permissions for a Print Queue – Administrative Client	The goal of this use case is to set permissions for a Print Queue, such as the priority of the Print Queue and the times it is available for shared use, as well as who may access it.
Submit a Print Job – Print Client	The goal of this use case is to print a document.
Manage Print Jobs – Print Client	The goal of this use case is for a User to manage his or her own submitted print jobs, including pausing them, resuming them, cancelling them, changing their priority, changing their order in the queue, or restarting them.

3.3.4 Systems Management Protocols Use Cases

[MS-GPSO]: Group Policy System Overview

Use case	Description
Apply Group Policy – GP Client	The goal of the use case is to retrieve policy content from the GP server and apply it on the GP client.
Administer Policy – Admin Tool	The goal of the use case is to create, update, and delete Group Policy content.

[MS-WMSO]: Windows Management System Overview

Use case	Description
Create a CIM Instance – Windows Management Client	The goal of this use case is to use the client to create a new CIM object in the CIM repository. The newly-created object will have attributes populated as specified by the client.
Invoke a Method on a CIM Instance – Windows Management Client	The goal of this use case is to invoke a method on a managed object.
Set Properties of an Instance – Windows Management Client	The goal of this use case is to set one or more properties of an existing CIM instance.
Query Properties of a CIM Instance – Windows Management Client	The goal of this use case is to retrieve the values of some set of properties of a CIM instance.
Monitor Events from WMS – Windows Management Client	The goal of this use case is to submit a query to WMS to notify the submitting user when conditions given in the query are met.
Delete CIM Object – Windows Management Client	The goal of this use case is to use the client to delete an existing CIM object in the CIM repository.
Attempt Delete on CIM Object – Windows Management Client	The goal of this use case is to illustrate the failure of a requested CIM object deletion.

3.3.5 Application Services Use Cases

[MS-MQSO]: Message Queuing System Overview

Use case	Description
Create or Modify Queue – Application	This use case is initiated by an application to create a queue or modify the properties of a queue.
Query Queue Information – Application	This use case is initiated by an application to query configuration and runtime information about the Message Queuing System.
Send Message to Queue – Application	This use case places a message in a queue.
Send Message in Transaction – Application	This use case places a message in a queue under the context of an atomic transaction.
Transfer Message – Application	This use case transfers a message from one queue manager to another.
Receive a Message from a Queue – Application	This use case receives a message from a queue.
Receive a Message in Transaction – Application	This use case receives a message from a queue under the context of an atomic transaction.

[MS-TPSO]: Transaction Processing Services System Overview

Use case	Description
Create a Transaction – Application	The goal of the use case is to start a new transaction with a transaction manager in the system.

Use case	Description
Create a Transaction – External Application	This use case is the same use case as Create a Transaction – Application, with External Application as the direct actor.
Do Transacted Work – Application	The goal of the use case is perform a set of operations within an atomic transaction.
Do Transacted Work – External Application	This use case is the same use case as Do Transacted Work – Application, with External Application as the direct actor.
Do Remote Transacted Work with Pull Propagation – Application	The goal of the use case is perform a set of operations in an atomic transaction on a remote resource, where the remote resource has a separate transaction manager. The transaction information is communicated to the remote transaction manager by using pull propagation.
Do Remote Transacted Work with Pull Propagation – External Application	This use case is the same use case as Do Remote Transacted Work with Pull Propagation – Application, with External Application as the direct actor.
Do Remote Transacted Work with Push Propagation – External Application	The goal of the use case is to perform a set of operations in an atomic transaction on a remote resource, where the remote resource has a separate transaction manager. The transaction information is communicated to the remote transaction manager by using push propagation.
Pull Transaction – Transaction Manager	The goal of the use case is to pull the information about a transaction from a remote transaction manager.
Pull Transaction – External Transaction Manager	This use case is the same use case as Pull Transaction – Transaction Manager, with External Transaction Manager as the direct actor.
Push Transaction – Transaction Manager	The goal of the use case is to push the information about a transaction to a remote transaction manager.
Enlist in a Transaction – Resource Manager	The goal of the use case is enlist a resource manager in a particular transaction. When it is enlisted, the resource manager will be able to participate in the coordination of the transaction.
Enlist in a Transaction – External Resource Manager	This use case is the same use case as Enlist in a Transaction – Resource Manager, with External Resource Manager as the direct actor.
Complete a Transaction – Application	The goal of the use case is to complete a transaction with an abort or a commit outcome.
Complete a Transaction – External Application	This use case is the same use case as Complete a Transaction – Application, with External Application as the direct actor.
Drive Completion of a Transaction – Transaction Manager	The goal of the use case is to inform the transaction participants of the outcome of the transaction and driving a consistent outcome.
Do Transaction Recovery – Resource Manager	The goal of the use case is to discover the outcome of transactions.
Do Transaction Recovery – External Resource Manager	This use case is the same use case as Do Transaction Recovery – Resource Manager, with External Resource Manager as the direct actor.

Use case	Description
Do Transaction Recovery – Transaction Manager	This use case is the same use case as Do Transaction Recovery – Resource Manager, with Transaction Manager as the direct actor.
Do Transaction Recovery – External Transaction Manager	This use case is the same use case as Do Transaction Recovery – Resource Manager, with External Transaction Manager as the direct actor.
Manage Transaction Managers – Management Tool	The goal of the use case is to monitor and administer a transaction manager in the system.
Manage Transactions – Management Tool	The goal of the use case is to monitor and administer a transaction manager in the system.

3.3.6 General Use Cases

The following tables summarize the use cases for the protocols that support collaboration, media services, protected content, remote desktop services (terminal services), and update services functionality.

[MS-CSSO] Collaboration Services System Overview

Use case	Description
Collaboration Client Registration - Client	The goal of this use case is to allow a Collaboration Client to publish its name and IP address in an ILS Server in order to allow a second Collaboration Client to look up the IP address of the first Collaboration Client.
Collaboration Client Query - Client	The goal of this use case is to query an ILS Server for a list of registered Collaboration Clients in order to select one for which to receive its location information.

[MS-MGSO]: DirectPlay System Overview

Use case	Description
Host a DirectPlay Game – Host Entity	The goal of the use case is to host a DirectPlay game session.
Join a DirectPlay Game – Client Entity	The goal of the use case is to join a DirectPlay game.

[MS-MSSO]: Media Streaming Server System Overview

Use case	Description
Publish Content to Media Server – Encoder	The goal of this use case is to create content for broadcast through the Media Streaming Server System.
Publish Secure Content to Media Server – DRM Packager	The goal of this use case is to provide protected content to the Media Server for streaming.
Stream Content from Media Server – Media Player	The goal of this use case is to stream media files and streams from Media Servers.
Request License from License Server – Media Player	The goal of this use case is to provide and enable playback for DRM protected media streams.
Log Statistics to Servers – Media Player	The goal of this use case is to obtain statistics for the Media

	Streaming Server System experience.

[MS-RMSO]: Rights Management Services System Overview

Use case	Description
Enroll RMS Server – RMS Server	The goal of this use case is to enroll the server with the Microsoft Cloud Service so clients trust the server and send it requests.
Bootstrap RMS Client – RMS Client Application	The goal of this use case is to prepare an RMS Client Application to participate in the RMS System.
Acquire Templates – RMS Client Application	The goal of this use case is to retrieve the rights policy templates published by the RMS Server for use in publishing protected content.
Publish Protected Content Online – RMS Client Application	The goal of this use case is to publish protected content by communicating directly with the RMS Server.
Publish Protected Content Offline – RMS Client Application	The goal of this use case is to protect content without making calls to an RMS Server.
Consume Protected Content – RMS Client Application	The goal of this use case is to remove protection from and consume content protected by RMS.

[MS-TSSO]: Terminal Services System Overview

Use case	Description
Establish a Connection to a TS Server in an Intranet Environment – RDP Client	The goal of this use case is for an RDP Client to establish a connection with a TS Server.
Establish a Connection to a VM Host in an Intranet Environment – RDP Client	The goal of this use case is for an RDP Client to establish a connection with a VM Host.
Establish a Connection Using a TS Gateway – RDP Client	The goal of this use case is for an RDP Client to establish a connection with a TS Server.
Establish a Connection to a TS Server in a TS Server Farm – RDP Client	The goal of this use case is for an RDP Client to establish a connection to a TS Server within a server farm.
Access Local Drives on an RDP Client – Remote Application	The goal of this use case is for the Remote Application to access local drives on the RDP Client.
Redirect Clipboard Data from a Remote Application – RDP Client	The goal of this use case is to use the local Clipboard of the RDP Client to perform Clipboard operations on a Remote Application running on a TS Server.
Use Printer on an RDP Client – Remote Application	The goal of this use case is for the Remote Application to send a print job to a printer on the RDP Client, which prints the job.
Redirect Smart Card Data from an RDP Client – Remote Application	The goal of this use case is for the Remote Application to access a Smart Card on the RDP Client.
Access Plug and Play Device on an RDP Client – Remote Application	The goal of this use case is for the Remote Application to access a Plug and Play device on the RDP Client.
Present Content from a TS Server on an RDP Client – Media Player	The goal of this use case is to present content streamed from the Media Player running on the TS Server to the RDP Client.
Access Audio Device on an RDP Client – Remote Application	The goal of this use case is for the Remote Application to access an audio device on the RDP Client.
Log Off from a Remote Session – RDP Client	The User of the RDP Client logs off from a TS Server, causing the user session on the TS Server to be closed.

Use case	Description
Disconnect from a Remote Session– RDP Client	The RDP Client disconnects from a TS Server, but the user session remains in a suspended mode for possible later use.

[MS-WSUSO]: Windows Server Update Services System Overview

Use case	Description
Configure Update Server — WSUS Administrator	The goal of this use case is to configure the update server according to the deployment requirements.
Manage Computer Groups — WSUS Administrator	The goal of this use case is to create computer target groups and establish membership of computers managed by the update server within those target groups.
Approve Update — WSUS Administrator	The goal of this use case is to approve an update to a target group for either installation or uninstallation. This is the primary use case that enables the system purpose of controlling update delivery.
Monitor Update Installation — WSUS Administrator	The goal of this use case is to generate update installation and applicability reports. How the system is implemented determines the type of reports that are generated.
Synchronize Server — WSUS Administrator	The goal of this use case is to synchronize a DSS in an update server hierarchy with updates and approvals from a USS.
Configure Update Client — Computer User	The goal of this use case is to assign user-specified values to configuration settings that control the client's behavior.
Start Update Scan — Computer User	The goal of this use case is to discover changes in the set of updates available to the client computer and their deployments since the last time the use case was executed. Additionally, update metadata is retrieved from the update server for new updates.
Install Updates — Computer User	The goal of this use case is to carry out the directive the WSUS Administrator specified for each update that is applicable to the client computer. These directives include Install and Uninstall. An update installation can also entail a download of update files.

4 System Context

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

4.1 System Environment

The individual system documents describe the relationship between each protocol system that are part of the Windows Protocols System and its environment. This section provides some additional information that applies to the environment of all Windows Protocols systems.

The Windows Protocols System environment is a distributed computing environment that is based on a client-server framework called the Distributed Computing Environment (DCE), which was specified by an industry consortium in the early 1990s. Typically, implementations of DCE are divided into administrative units called "cells" with the following core services available to all clients and servers that participate in a given cell:

- Remote Procedure Call (RPC)
- Authentication Service
- Threads Service
- Distributed Directory Services
- Distributed Time Services (DTS)

The Windows NT-based operating systems are based on the DCE framework. In Windows, the term "domain" corresponds to the term "cell" and the DCE core services are still pervasive in Windows today.

Computers that are members of a domain (domain members) include the following key services:

- Remote Procedure Call [MS-RPCE]
- An Authentication Service based on Kerberos [MS-KILE] and [MS-AUTHSO]
- A Distributed Directory Service based on Active Directory [MS-ADTS] and [MS-DISO]
- Time Services [MS-W32T]

In a Windows environment, there is no inherent need to add threads as a separate service because the operating systems provide this service. There are also additional services such as the Distributed File System intended to support user functionality. This framework has been extended and improved over time. For example, HTTP-based distributed computing is now common place and SOAP-based protocols are often used instead of traditional RPC. Despite this, the core DCE model remains.

The Windows environment has many optional features. A naming service, for example, is highly desirable but not essential. IPv6 is also desirable but not essential. The minimum environment required for hosts is that there is IPv4 connectivity. More complex, secure and robust configurations have additional requirements such as naming services and time services.

Many of the Windows protocols require a durable storage system to maintain the state of the directory and meet the transactional guarantees specified in [MS-ADTS] section 3.1.1.5.1.3. This storage system SHOULD provide a means of securing the contents of the storage system from unauthorized access.

Many protocols also require a networking system that clients can be use to send requests to the directory server and receive a response. This networking system MUST support the TCP, UDP, and Server Message Block (SMB) transports. This networking system MUST also supply access to the Domain Name System

(DNS). DNS can be used by clients of the Active Directory System in order to locate directory servers using the algorithms described in [MS-ADTS] section 7.3.

Several Windows protocols are RPC-based and depend on the availability of an RPC runtime that implements an RPC mechanism as described in [MS-RPCE].

The environment for Fax Services, the Print Services System, and the File Access Services System protocols exist within Windows as sets of drivers and services. This functionality is consumed by other Windows Protocols System services and applications. Users are not expected to interact directly with these services, but rather indirectly, through text or graphical shells, applications, and higher level Windows APIs.

Specifically for the protocol described in [MS-FAX] enables communication between a client system and a FAX Server. Use of the FAX protocol is completely distinct from the use of the printing-related protocols described in the Print Services System documented in [MS-PSSO]. Although a single fax driver appears locally in the user interface on a client system (enumerated as a printer driver), no [MS-PSSO] protocols are used to interact with that driver or to support any FAX use cases. The fax printer driver uses only the [MS-FAX] protocol to transfer faxes from the client system to a FAX Server, without relying on any member protocol of the Print Services System.

4.2 System Assumptions and Preconditions

All Windows protocols assume that the following preconditions are true when any of the Windows Protocol Family and Defined Tasks systems are started. These systems assume that the precondition is established before the system in question is started and that it is verified outside of the system. The Windows protocols, protocol families, and Defined Task protocols do not verify these preconditions.

The preconditions are as follows:

- There is a network that provides a viable transport for communications between the directory server, and its clients MUST be available. As specified in the previous section, this network MUST supply access to DNS.
- The transport protocol for that network MUST be available and configured (for example, the TCP transport must be configured with a valid IP address).
- A security infrastructure that supports identity, authentication, and authorization is in place.
- The Active Directory System as specified in [MS-ADSO] is the only system that assumes the server is configured as a domain controller.
- Most of the protocols from the Windows Protocol System can be implemented with a client computer that is either joined to a domain or is a standalone computer. For protocol implementations that require or use the services provisioned by a domain controller or consume the authentication services documented in [MS-AUTHSO], the following preconditions apply:
  - A computer in the domain needs to be promoted and configured as the primary domain controller for that domain to support the domain and authentication services documented in [MS-DISO] and [MS-AUTHSO].
  - The user account for all domain clients needs to be created and provisioned on the domain controller by the domain administrator.
  - Domain clients must have successfully joined the domain that provisions domain services for them.
- The following systems assume the client computer is joined to a domain:

- Authentication Services System [MS-AUTHSO]
- Certificate Autoenrollment System [MS-CAESO]
- Domain Interactions System [MS-DISO]
- File Access Services System [MS-FSSO]
- Group Policy System [MS-GPSO]
- Message Queuing System [MS-MQSO]
- Network Access Protection System [MS-NAPSO]
- Print Services System [MS-PSSO]
- Rights Management Services System [MS-RMSO]
- Terminal Services System [MS-TSSO]
- Windows Management System [MS-WMSO]

- As part of the Windows startup procedure for the domain clients, the following preconditions apply:
  - Successful connection to network and startup of networking stacks.
  - Network discovery of site and domain controller.
  - Establishment of a secure channel with domain controller.
  - Download of Group Policy settings.
  - Client certificate autoenrollment.
  - Time synchronization.
  - Dynamic DNS update.
- The following systems require that the Authentication Services System [MS-AUTHSO] is available:
  - File Access Services System [MS-FSSO]
  - Print Services System [MS-PSSO]
- The following systems require that the Active Directory System [MS-ADSO] is available:
  - File Access Services System [MS-FSSO]
  - Message Queuing System [MS-MQSO]
  - Print Services System [MS-PSSO]
  - Certificate Autoenrollment System [MS-CASO]
- For protocols of the Windows Protocols System that use either HTTP [RFC2616] or HTTPS [RFC2818] as a transport, the HTTP server has to be configured and the HTTP service should be started so that the HTTP server can respond to HTTP requests from the client.

- For protocols that use SMB as a transport or depend on SMB for file access, the SMB service must be properly configured and started.
- For protocols that use RPC as a transport, the RPC service on both the client and server computers must be properly configured and started.
- For protocols that depend on DCOM as a transport or to authenticate its clients.

The following systems assume there is durable storage to store system state:

- Active Directory System [MS-ADSO]
- Print Services System [MS-PSSO]
- Rights Management Services System [MS-RMSO]
- Transaction Processing Services System [MS-TPSO]
- Windows Server Update Services System [MS-WSUSO]

4.3 System Relationships

This section describes the relationships across the system and external components, system dependencies, and other systems influenced by this system.

4.3.1 Black Box Relationship Diagrams

The individual system documents describe the black box relationships for each Windows Protocols System. This section provides two additional black box relationship diagrams that provide a composite view of the black box diagrams for protocols and systems in the File, Print, and Fax protocols group and the Systems Managements protocols group.

4.3.1.1 Black Box Relationship for the File, Print, and Fax Protocols

The following black box diagram of the File Access Services System and the Print Services System illustrates interfacing external system and service components.

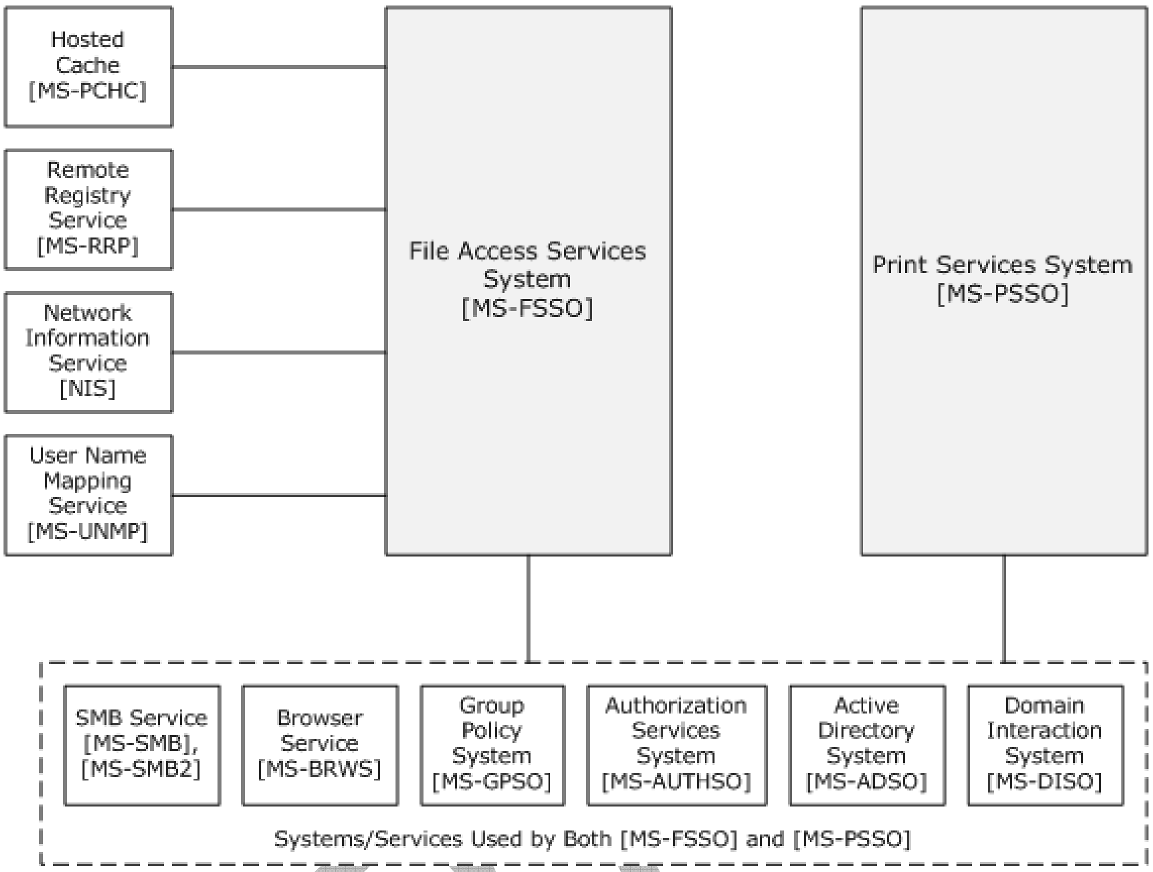

Hosted Cache [MS-PCHC]
Remote Registry Service [MS-RRP]
Network Information Service [NIS]
User Name Mapping Service [MS-UNMP]
File Access Services System [MS-FSSO]
Print Services System [MS-PSSO]
SMB Service [MS-SMB], [MS-SMB2]
Browser Service [MS-BRWS]
Group Policy System [MS-GPSO]
Authorization Services System [MS-AUTHSO]
Active Directory System [MS-ADSO]
Domain Interaction System [MS-DISO]
Systems/Services Used by Both [MS-FSSO] and [MS-PSSO]

Figure 12: Black box for File Access Services System and Print Services System

The following components are shared by both systems:

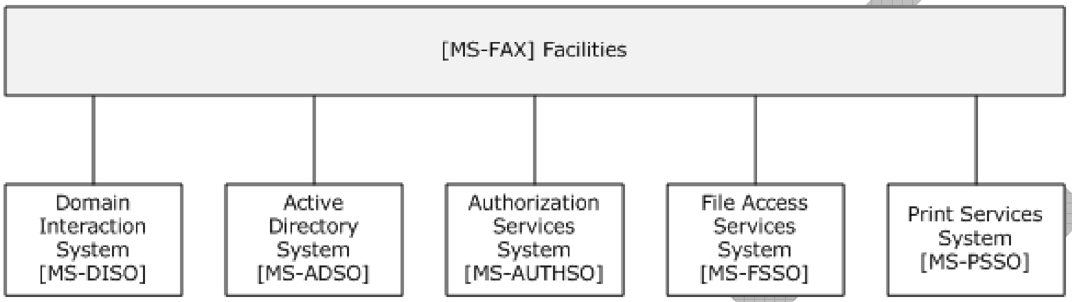
- SMB Service, supporting print functionality using the SMB Protocol Family, including [MS-SMB], [MS-SMB2], and [MS-CIFS].
- Browser Service, [MS-BRWS],supporting the publication of shared print resources in workgroup environments.
- Group Policy System, [MS-GPSO], supporting the pushing of preconfigured connections between Print Servers and Print Clients.
- Authorization Services System, [MS-AUTHSO],supporting domain-based security.
- Domain Interaction System, [MS-DISO],supporting domain-based security.
- Active Directory System, [MS-ADSO],supporting the publishing of shared print resources.

In addition, there is a mutual dependency between the Print Services System and the File Access Services System, as described in section 3.1.4.1.

The following system and service components interface to the File Access Services System, each used as described in [MS-FSSO]:

- Hosted Cache, [MS-PCHC]
- Remote Registry Service, [MS-RRP]
- Network Information Service, [NIS]
- User Name Mapping Service, [MS-UNMP]

The Black Box diagram of the [MS-FAX] Facilities below illustrates interfacing external systems. All of these systems are accessed using local API calls on client and server computers implementing [MS-FAX] support, not by using system protocols defined by the external systems.



[MS-FAX] Facilities
Domain Interaction System [MS-DISO]
Active Directory System [MS-ADSO]
Authorization Services System [MS-AUTHSO]
File Access Services System [MS-FSSO]
Print Services System [MS-PSSO]

Figure 13: Black Box for [MS-FAX] Facilities

Of the supporting systems, the Print Services System is the most directly used by [MS-FAX] facilities: the Print Spooler is used for enumerating available Fax Queues to which a client can connect, and enabling a client to connect to a Fax Queue.

4.3.1.2 Black Box Relationship for the Systems Management Protocols

This diagram shows, at a high level, the interaction of components of the Windows Management Capabilities. The Group Policy System and the Windows Management System together provide complete monitoring and control of every hardware and software resource in the managed system. The Group Policy System is used to control access permissions to resources and their management metadata. The Windows Management System provides access to the resources and their management metadata, IF the entity requesting that access has the proper access permissions. The Group Policy System allows system administrators to collect multiple users or system resources into one named security principal in order to make system administration much easier. For more details on the capabilities of the Windows Management System, see [MS-WMSO]. For more details on the capabilities of the Group Policy System, see [MS-GPSO].

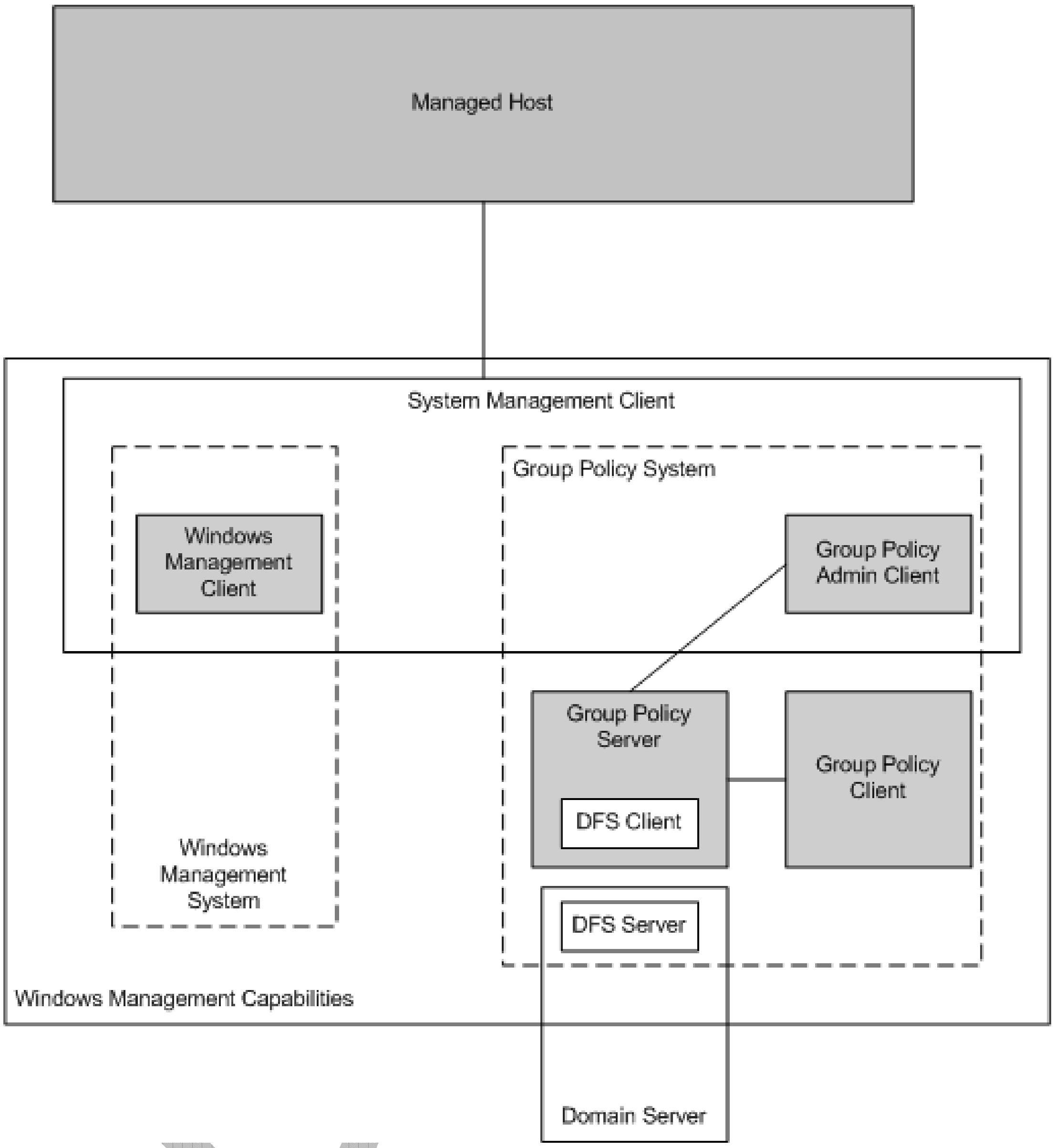


Figure 14: Windows Management Capabilities black box diagram

4.3.2 System Dependencies

The individual system documents describe system dependencies for each Windows protocols system. There are no additional Windows-wide system dependencies.

4.3.3 System Influences

The individual system documents describe system influences for each Windows Protocols System. There are no additional Windows-wide system influences.

4.4 System Applicability

This system is applicable for client to server usage. This system does not describe server to server usage, such as usage within an enterprise.

The Technical Documents (TDs) for the protocols describe the applicability of the individual protocols (see section 1.6, "Applicability Statement") and Protocol Family System Overviews give the applicability of their protocol family as described in the following sections.

4.5 System Versioning and Capability Negotiation

The TD for each protocol describes the available vendor-extensible fields (see section 1.7, "Versioning and Capability Negotiation"). Protocol Family System Overviews provide the vendor-extensible fields of their protocol family (see section 4.5). The available versions of the system will be based on the collected versions and capabilities negotiated with and among the individual protocols.

There is no versioning or capability negotiation at the level of the entire Windows Protocols System. There are standard bundles of versions and capabilities with implementation-specific releases.

4.6 System Vendor-Extensible Fields

The TD for each protocol describes the available vendor-extensible fields (see section 1.8, "Vendor-Extensible fields"). Protocol Family System Overviews give the vendor-extensible fields of their protocol family as described in this section.

Certain vendor-extensible fields are shared across multiple TDs. If you define a value for one protocol, you should not associate a different meaning for the same value in another protocol.

The most common vendor-extensible fields include the following:

- **HRESULTs**: Vendors can choose their own values, as long as the **C** bit (0x20000000) is set, indicating it is a customer code. For more details, see [MS-ERREF]. The structures documented in [MS-ERREF] have no vendor-extensible fields.
- **Win32 Error Codes**: Vendors should reuse those values with their indicated meanings. Choosing any other value runs the risk of a collision in the future. For more details, see [MS-ERREF].
- **NTSTATUS**: Vendors can choose their own values for this field, as long as the **C** bit (0x20000000) is set, indicating it is a customer code. For more details, see [MS-ERREF].
- **SOAP Fault and SOAP Fault Codes**: SOAP Fault detail elements can be vendor selected when the content of the SOAP Fault Detail element does not affect interoperability of the protocol. SOAP Fault Codes can be vendor selected when the choice of the SOAP Fault Code does not affect interoperability of the protocol. For more details about SOAP Fault, see section 4.4 in [SOAP1.1] 4.4 and for more details about SOAP Fault Codes see section 5.4 in [SOAP1.2/1].
- Vendor extensibility in an XML Schema is indicated by ##OTHER or ##ANY for XML attributes or elements in their own namespace.
- Vendor-defined objects should be unique across all the protocols using the object. Vendors extend the directory by adding objects, including schema objects as specified in [MS-ADTS] to control the vendor objects.

LDAP is not extensible by Active Directory applications. Applications extend the **directory** by adding objects, including **schema objects** to control the application objects.

5 System Architecture

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

5.1 Abstract Data Model

The Abstract Data Model for the Windows Protocols System can be found in the Abstract Data Model sections of the Systems Documents in the Normative Reference section of this document. There are two Systems Documents whose Abstract Data Models do not have elements that are used in any other Windows Protocols System. Those documents are Multi-player Games [MGSO], where the user identities are not tied to a specific domain, and Media Server System [MSSO], again where the user identities are not tied to a specific domain.

The following Windows Protocols System abstract data model elements are common across the Windows Protocol Family Systems and Defined Task Systems:

- User Identity detailed in [ADSO]
  - Name - (maps to Active Directory cn attribute) defined in [ADSO].
- User Credentials detailed in [AUTHSO]
  - UserClientMachineServiceTicket – The Kerberos service ticket for the member machine.
  - UserTGT – The Kerberos ticket-granting ticket (TGT) for user logon.
  - UserGroupPolicy – Group policy to be applied to the user during logon.
  - UserDFSRef – DFS referral information for the user.
- Group Policy elements detailed in section 5.3.1 in [MS-GPSO]
- Domain names
- UNC identifiers

The following Abstract Data elements are not defined in System Documents, as their use is ubiquitous throughout the Internet:

Computer Name: a **Domain Name System (DNS)** name or a **NetBIOS** name. For additional information about DNS, see [MSFT-DNS]; for NetBIOS, see [RFC1001] and [RFC1002].

File Name: A network identifier used to uniquely identify a file. Defined in [MSDN-FILE].

5.2 White Box Relationships

The individual system documents describe the white box relationships for each Windows protocols system. This section provides two additional white box relationship diagrams that provide a composite view of the black-box diagrams for protocols and systems in the File, Print, and Fax protocols group and the Systems Managements protocols group.

5.2.1 File, Print, and Fax Protocols

The diagram in this section shows how system components can be distributed across multiple computers. The diagram illustrates the communications that occur between the File Access System and the Print

Services System, as well as communications to external systems ([MS-DISO], [MS-AUTHSO], [MS-ADSO], and [MS-GPSO]. The diagram also shows internal communications that use protocol-based communications between clients and servers implementing these two systems, and communications using local API calls within a client and a server implementing both systems.

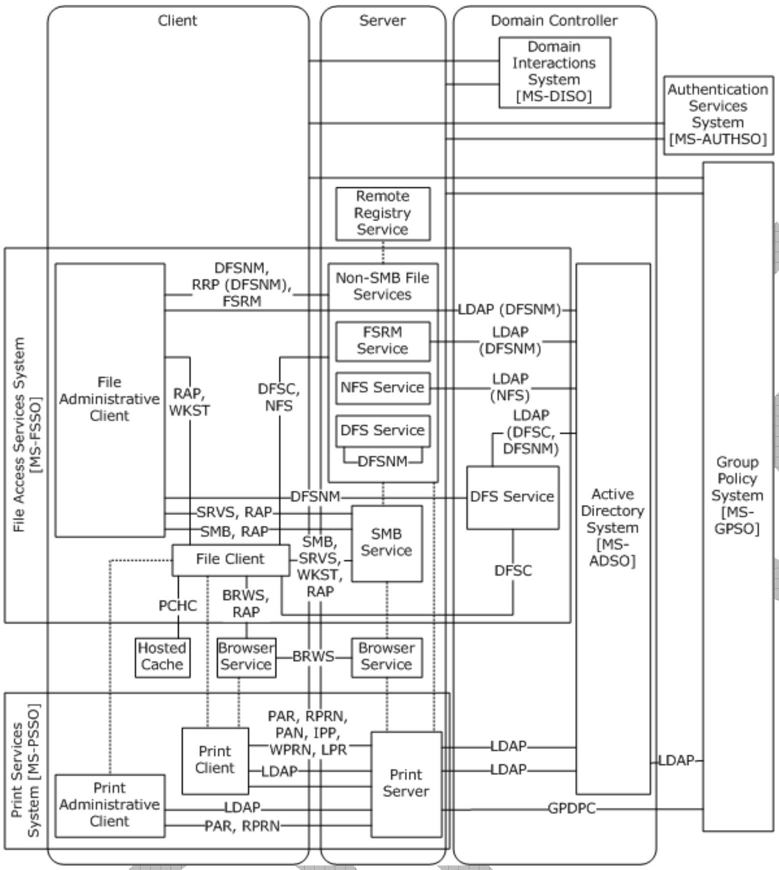


Figure 15: White Box showing File Access Services System and Print Services System interactions

(Dotted lines indicate local calls; solid lines indicate protocol-based communications.)

The White Box diagram for [MS-FAX] facilities shows the use of the [MS-FAX] protocol between the client and Fax Server, as well as the component interactions between the components of [MS-FAX] facilities and their use of local calls to perform fax-related functions, and the external systems that support the fax clients and fax servers ([MS-DISO], [MS-ADSO], and [MS-AUTHSO]).

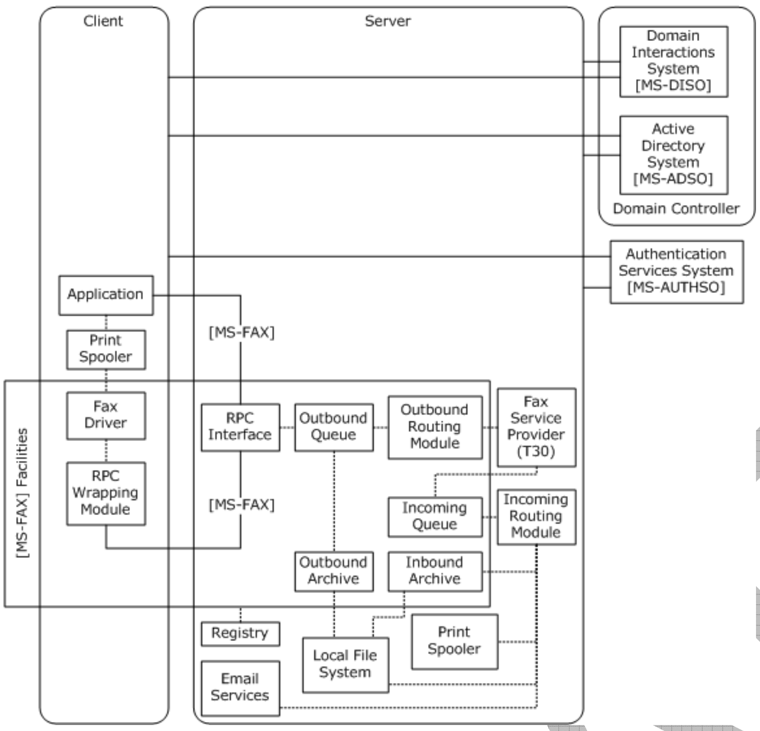


Figure 16: White Box showing [MS-FAX] facilities

(Dotted lines indicate local calls; solid lines indicate protocol-based communications.)

5.2.1.1 Protocol Layering for File, Print, and Fax

The protocol layering diagram in this section shows the mutual use of various protocols to perform File-related, Print-related, and Fax-related functions.

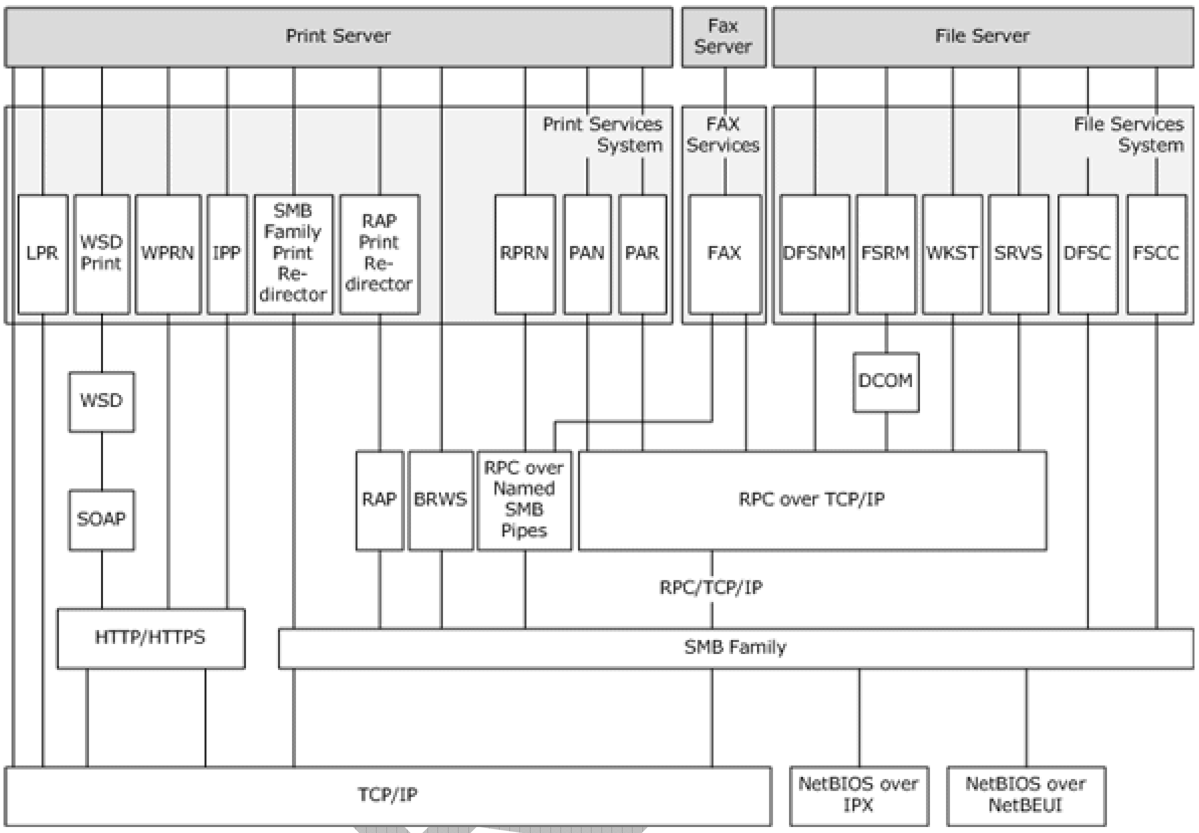


Figure 17: Protocol Layering diagram

The Print Services System includes protocol stacks that support HTTP/HTTPS for internet printing scenarios, as well as RPC-based protocols for network printing.

The [MS-FAX] protocol uses RPC/SMB named pipes to send fax data to the server, and RPC/TCP/IP to send fax-related events to the client.

The File Access Services System uses the protocol layering noted in the diagram, as well as NFS-related protocols not shown here, but illustrated in detail in [MS-FSSO] section 5.2.

5.2.2 System Management Protocols

There are two major components of the Systems Management Protocols: the Windows Management System, discussed in [MS-WMSO], and the Group Policy System, discussed in [MS-GPSO]. The following diagrams show the relationships between the two systems.

5.2.2.1 The Windows Protocols System Management System

The Windows Protocols System Management System provides management capabilities over large and widely distributed sets of computers.

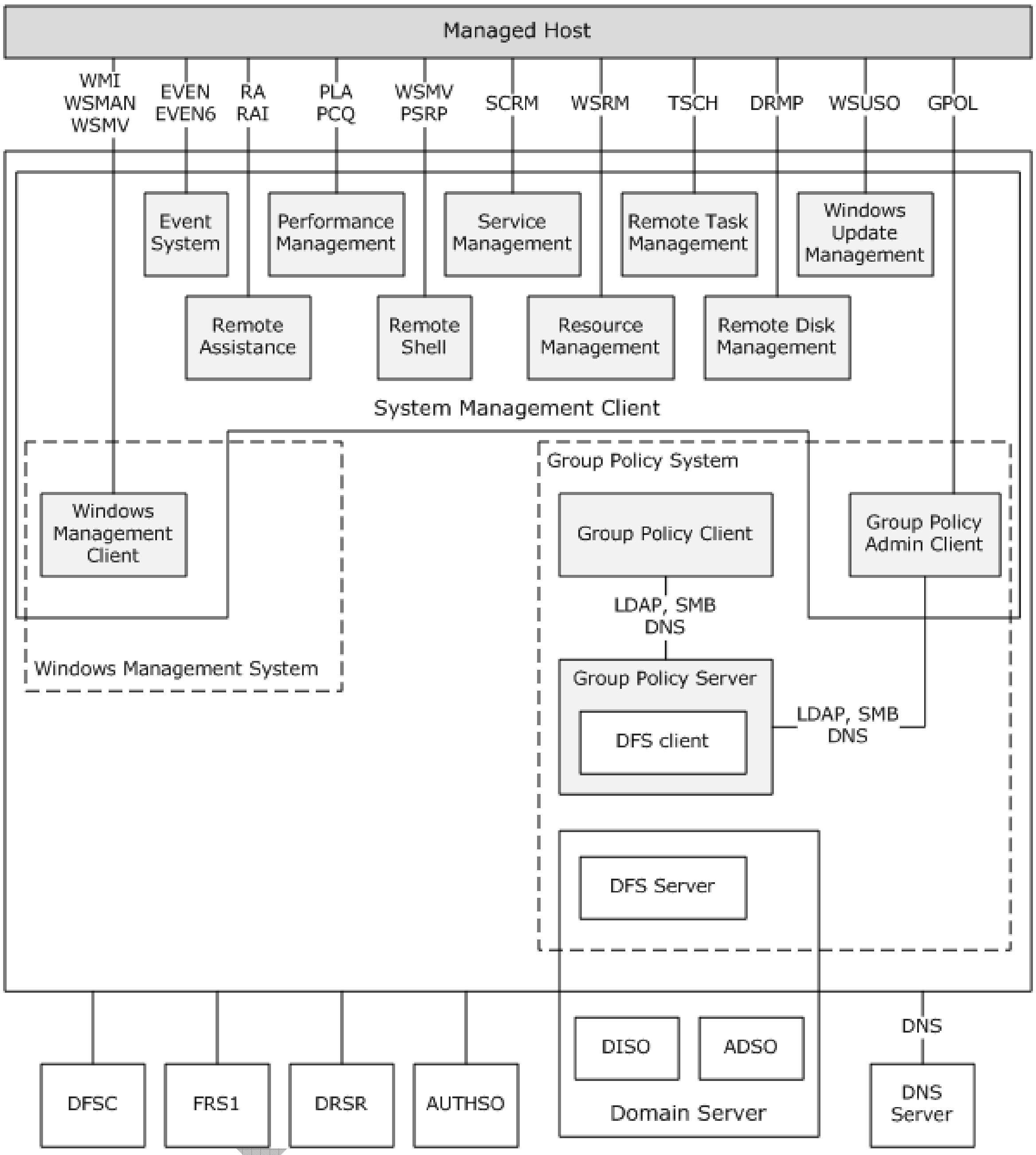


Figure 18: WMSO GPSO Interrelations White Box Diagram

The Windows Management System components shown in the previous figure are covered in more detail in [MS-WMSO]. The Group Policy System components shown in the previous figure are covered in more detail in [MS-GPSO]. Components which are not discussed in either of these documents are detailed as follows.

System Management Client

- In addition to the Group Policy Admin Client and the Windows Management Client, the System Management Client allows access to and modification of other Windows management components.
- The Event System allows access to event logs maintained by the system. An expanded white box detailing the event system and its protocols is shown below.
- Remote Assistance allows an administrator to share the desktop view of a remote computer.
- Performance Management allows examination of performance logs maintained by the system.
- Remote Shell capabilities allow an administrator to open a shell on a remote computer. A Remote Shell scenario utilizing the PowerShell protocol can be found in section 7.1.
- Service Management allows the remote administration of Windows Services.
- Resource Management allows the management of machine resources such as processor allocation, memory allocation, and so on.
- Remote Task Management allows the scheduling of tasks to be run on remote computers.
- Remote Disk Management allows the management of disk resources such as volume names and sizes on remote computers.
- Windows Update Management allows the configuration of the Windows Update feature, determining when and which Windows Updates are downloaded and installed on the local computer.

Domain Server

The Domain Server provides domain membership and management capabilities for Windows domains.

Supporting Systems

- DFCS: The Distributed File System (DFS): Referral Protocol allows file system clients to resolve names from a namespace distributed across many servers and geographies into local names on specific file servers.
- FRS1: The File Replication Service protocol allows files to be replicated to remote locations while retaining the same UNC.
- DRSR: The Directory Replication Service (DRS) Remote protocol allows Active Directory entries to be replicated to local servers, for enhanced access speeds.
- AUTHSO: The system described in [MS-AUTHSO] handles user credentials, authentication, and authorization to system resources.
- DNS Server: The DNS server provides resolution for Internet domain names.

5.2.2.2 The EventLog system

The EventLog system is the only complex system component shown in the previous diagram that does not have an associated System Document. The following diagram shows the internal white box architecture for the EventLog system.

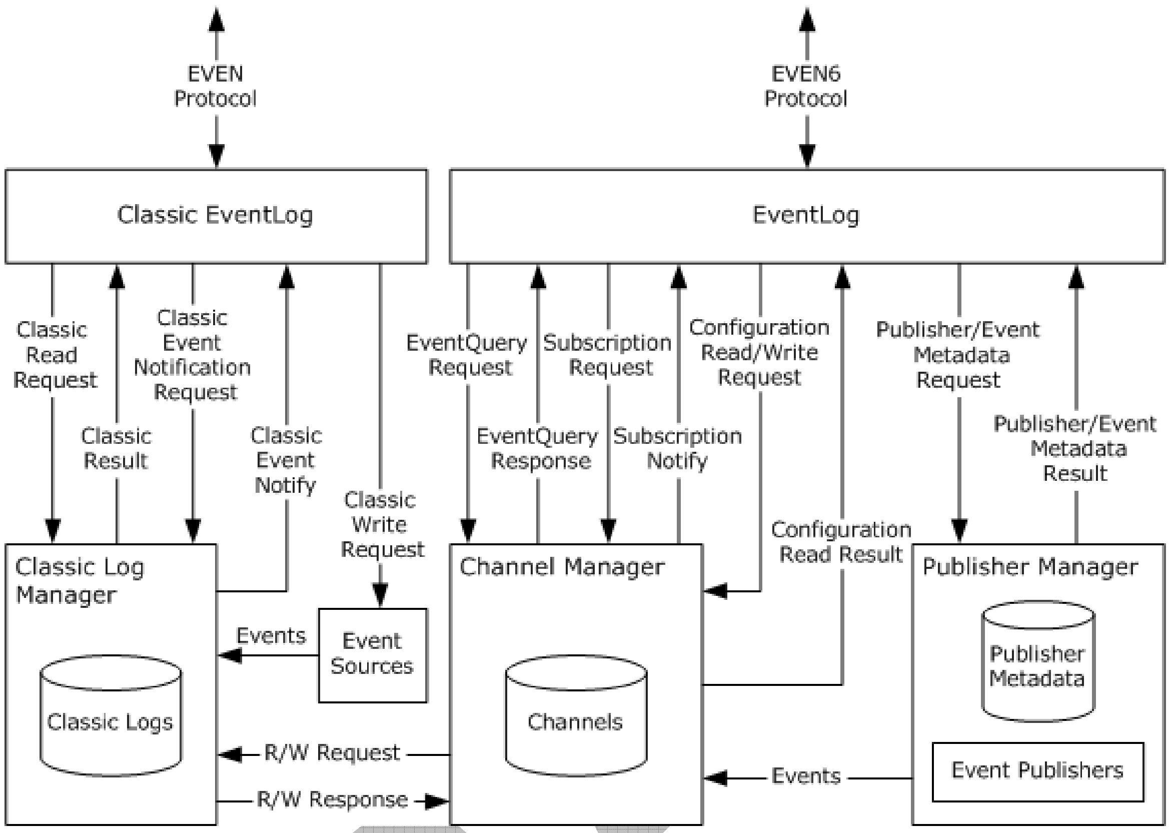


Figure 19: EventLog System white box relationships

The EVEN protocol and the EVEN6 protocol provide access to EventLog data, although the protocols access different sets of data. The data available through the EVEN protocol is a strict subset of the data available through the EVEN6 protocol. As shown in the diagram, the Publisher Manager used in EVEN6 provides Publisher Metadata that is not available through EVEN. In addition, the EVEN protocol allows events to be written to the Classic Log Manager, but EVEN6 does not provide a write capability to the System Management Client. For more details on the EVEN and EVEN6 protocols and accessible data, see [MS-EVEN] and [MS-EVEN6].

6 System Details

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

6.1 Architectural Details

6.1.1 System Document Examples

This section lists the examples (also referred to as scenarios) from each System Document and provides a short description of each example.

6.1.1.1 Networking and Transports Protocols

[MS-NAPSO]: Network Access Protection System Overview

Example	Description
NAP Client Creating a Statement of Health (SoH)	This example illustrates a NAP (Network Access Protection) Client creating a SoH. The client utilizes several SHA, NAP Agent and EC functions to accomplish the request.
NAP Client Sending a SoH	This example illustrates a NAP Client sending a SoH. The client utilizes several NAP Agent and EC functions to accomplish the request.
PDP (NAP health policy server) Receiving a SoH.	This example illustrates a PDP receiving a SoH. The PDP utilizes several Policy Engine, NAP Validator and SHV functions to accomplish the request.
PDP (NAP health policy server) Evaluating Health	This example illustrates a PDP evaluating health. The PDP utilizes several Policy Engine, NAP Validator and SHV functions to accomplish the request.
PDP (NAP health policy server) Creating a Statement of Health Response (SoHR)	This example illustrates a PDP creating a SoHR. The PDP utilizes several Policy Engine, NAP Validator, and SHV functions to accomplish the request.
PDP (NAP health policy server) Sending a SoHR.	This example illustrates a PDP sending a SoHR. The PDP utilizes several Policy Engine, NAP Validator and SHV functions to accomplish the request.
PDP (NAP health policy server) Enforcing Network Restrictions	This example illustrates a PDP enforcing network restrictions on a client. The PDP utilizes PEP and the NAP Client functions to accomplish the request.
NAP Client Receiving an SoHR	This example illustrates a NAP Client receiving a SoHR. The client utilizes several NAP Agent, SHA and EC functions to accomplish the request.
NAP Client Processing an SoHR	This example illustrates a NAP Client processing a SoHR. The client utilizes several NAP Agent, SHA and EC functions to accomplish the request.
NAP Client remediating Itself	This example illustrates a NAP Client remediating itself. The client utilizes several NAP Agent, SHA, and EC functions to accomplish the request.

6.1.1.2 Identity and Security Protocols

[MS-ADSO]: Active Directory System Overview

Example	Description
Create a User Using the LDAP Protocol	In this scenario, an administrator creates a user account via the LDAP protocol. For illustrative purposes, the Active Directory Users and Computers (ADUC) administration tool is used to perform this operation.
Create a User via SAMR	This scenario uses the SAMR protocol rather than LDAP to create the user and only provides a minimal set of attributes for the newly created account.
Create a User via LDAP – Error Due to Insufficient Access Rights	This scenario shows how the directory service responds when the creator does not have sufficient access rights. This scenario uses the LDAP protocol to attempt the user creation, although it could have used the SAMR protocol instead.
Change a User's Password via LDAP	A common administration task is to change the password for an existing user. One way of accomplishing this is to use the LDAP protocol to communicate with the Active Directory System.
Change a User's Password via LDAP – Error Due to Insufficient Password Complexity	In this scenario the new password specified for the user fails to meet the domain's password policy requirements and the directory server does not update the user's password.
Group Membership	This scenario builds on the previous scenarios as it uses the LDAP protocol to fetch and modify group membership and then uses the SAMR protocol to retrieve the updated group membership.
Query Users via the Web Services Protocols	To perform this task the client runs a program that sends a query to the directory service using the Web Services protocols. This scenario uses the WS-Enumeration protocol to communicate with the directory service.

[MS-AUTHSO]: Windows Authentication Services System Overview

Example	Description
Interactive Domain Logon	Interactive logon can be performed in a number of ways. This example shows the password-based Kerberos exchanges.
HTTP Web Access Authentication	This example provides the steps that the server undertakes for HTTP Web Access Authentication. The example details the protocol traffic between the HTTP client, HTTP server, and domain controller when the HTTP client is using NTLM authentication.
File System Services - SMB Server	This example describes the steps that the server undertakes to provide support for authentication of SMB file system services.
Remote Procedure Services - RPC Server	This example describes the steps that the server undertakes to provide authentication for RPC system services.

[MS-CAESO]: Certificate Autoenrollment System Overview

Example	Description
Autoenrollment in the Standalone Environment with XCEP/WSTEP Protocols	The goal of this example is to enroll for a certificate in an automated way. In this example, the computer on which the task executes is not joined to any domain. It has also been configured with policy server end point information in its local configuration.
Autoenrollment in the Domain Environment with CRTD/WCCE Protocols	The goal of this example is to enroll for a certificate in an automated way. In this example, the computer on which the task executes is joined to a domain. Also, a CA (WCCE server) exists in the same domain. Finally, a certificate template and Group Policy enabling autoenrollment have been configured by the domain administrator.

[MS-CASO]: Certification Authority System Overview

Example	Description
Enrollment from a Standalone CA (Basic Enrollment)	The simplest case of certificate enrollment is basic enrollment. In this example, the caller creates a PKCS#10 request by populating its fields as the caller chooses. The caller then uses an implementation that has a WCCE client component to submit the request to the WCCE server (CA).
Enrollment from an Enterprise CA (Template Based Enrollment)	This example builds on the example Enrollment from a Standalone CA (Basic Enrollment) by introducing an enterprise CA.
Enrollment with CA Administrator Approval	This example builds on the example Enrollment from an Enterprise CA (Template Based Enrollment) by introducing a CA Administrator who modifies and approves the certificate request before the certificate is issued.
Enroll on Behalf of Request and Renewal	This example builds on the example Enrollment from an Enterprise CA (Template Based Enrollment) by introducing a cosigner for the certificate request.
Private Key Archival and Recovery	This example builds on the example Enrollment from an Enterprise CA (Template Based Enrollment) by introducing private key archival and recovery.
Certificate Revocation	This example builds on the example Enrollment from an Enterprise CA (Template Based Enrollment) by adding the process of revoking a previously issued certificate.
Certificate Denied by Policy Algorithm	This example builds on the example Enrollment from an Enterprise CA (Template Based Enrollment), however, in this scenario, the caller requests a certificate based on a template for which it does not have permission to enroll. In this case the CA's policy algorithm denies the request after examining the permissions applied to the template.
Certificate Denied Due to Out-of-Sync Certificate Templates	This example builds on the example Enrollment from an Enterprise CA (Template Based Enrollment) and illustrates a situation where two Active Directory servers are out of sync, resulting in a version mismatch between the certificate templates used by client and server.

[MS-DISO]: Domain Interactions System Overview

Example	Description
Locating a Domain Controller	This example shows the pattern for locating a domain controller based on the domain name provided, both flat, NetBIOS names as well as the fully qualified Domain Name System (DNS) names preferred for AD-style domains.
Joining a Domain with a Predefined Account	This example illustrates the process by which a computer joins a domain using an account that is already configured within the domain.
Joining a Domain by Creating an Account via SAM	This is an example of a more secure variant for joining a domain that involves establishing the account on the domain controller via the SAM RPC interface and creating a new, random password for the domain client and the account in the domain.
Joining a Domain by Creating an Account via LDAP	This example illustrates the process by which a computer joins a domain after creating an account within the domain. This example creates the account via LDAP rather than using the SAM RPC interface.
Removing a Client from the Domain	This example illustrates the process in which a client is removed from the domain. The client must first be confirmed as part of the domain, then update information on the machine object so accompanying domain servers will detect that the client is being removed.

6.1.1.3 File, Print and Fax Protocols

[MS-FSSO]: File Access Services System Overview

Example	Description
Configure a File Service	Configure File Service is illustrative of typical Admin interaction with the File Access Services System as the Admin provisions remotely accessible storage on a given file server.
Find a File in a Workgroup	Find File in Workgroup is illustrative of typical User interaction with the File Access Services System as a User locates a file server and a share on it, obtains a directory listing, and then opens a file.
Find a File in a Domain	Find File in Domain is illustrative of typical User interaction with the File Access Services System as the User locates a file within a DFS namespace and then opens the file.
Two Applications Communicate using a Shared File	The sequence described in this example allows two applications, one using the NFS File Access Protocol and another using the SMB File Access Protocol to share a file on a remote file server.

[MS-PSSO]: Print Services System Overview

Example	Description
Discover and Utilize a Print Queue in a Domain	In this example, a Print Client locates a Print Queue in a domain, establishes a connection to the Print Queue, downloads a print driver, registers for notifications of printing status, submits a print job,

Example	Description
	receives notifications, and then unregisters for notifications and disconnects from the Print Queue. The LDAP protocol is used to discover the available shared Print Queues, after which the protocol described in [MS-RPRN] as well as the SMB family protocols are used for subsequent operations.
Discover and Utilize a Print Queue in a Workgroup	In this example, a Print Client running Windows XP and located in a workgroup locates a shared Print Queue on another computer in the workgroup, establishes a connection to the Print Queue and registers for notifications, submits a print job to the Print Queue and receives notifications, and unregisters for notifications and disconnects from the Print Queue.
Locate Print Queue in a Domain and Establish a Connection, then Submit a Print Job to a Manual Duplex Printer Using [MS-RPRN] and [MS-PAR] and Enable Unidirectional IHV-defined Communication Between Print Client and Print Server Using [MS-PAN]	In this example, a Print Client locates a Print Queue in a domain, establishes a connection to the Print Queue, downloads a print driver, enables OEM-defined communications between the Print Client and Print Server, submits a print job, and receives notifications.
Enumerate Print Jobs from All Users, Then Cancel Several Print Jobs	In this example, an administrator with appropriate privileges using an Administrative Print Client views and overrides print jobs submitted to a shared Print Queue by other users.
Provision a Print Queue Using [MS-RPRN] from an Administrative Client, then Delete the Same Print Queue Using [MS-PAR] from a Different Administrative Client	In this example, two Administrative Print Clients are involved in managing Print Queues: the first client provisions a Print Queue using the protocol described in [MS-RPRN] and the second client subsequently deletes the same Print Queue using the protocol described in [MS-PAR].
Send a Print Job to an SMB Share	In this example, a user performs commands from the command line, sending a print job to an SMB share by using the command "copy /b file \\server\printer" at the command line. The Print Client subsequently uses the SMB protocol family to copy the file to the specified SMB print share of the Print Server.

6.1.1.4 Systems Management Protocols

[MS-GPSO]: Group Policy System Overview

Example	Description
Populating Administrative Tools with Configuration Data	This example demonstrates the process that occurs when the Group Policy administrative tools load and retrieve the appropriate data from the stores that contain policy data.
Authoring a New Policy	This example describes the message flow during the action of authoring a new policy.
Administrative Tool Cannot Connect to a Domain Controller	When the Admin tool launches, it attempts to contact a domain controller (DC). That DC can be any available Read/Write DC in the domain to be managed. If a DC cannot be reached, the Admin tool generates an error message that states that the directory cannot be located.
Querying Active Directory for Scope of Management (SOM) and Version	In this example, a GP Client queries a GP Server for Scope of Management (SOM) and version information.

Example	Description
Information	
Client Applying Policy	The Client's interaction with the GP Server in policy application exhibits a pull application in which the Client polls a GP server to check for new user GPOs.
Client Cannot Connect to a Domain Controller When Applying Policy	When the Client reapplies policy, it attempts to contact a domain controller (DC). That DC can be any available readable DC in the domain. If a DC cannot be reached, the Client generates an error message to the event log that states that the directory cannot be located.

[MS-WMSO]: Windows Management System Overview

Example	Description
Single Request/Response Operations	A majority of the operations that are carried out within the WMS System are comprised of a single request and response. This example illustrates the communication flow of these simple operations.
Enumerations	When enumerating a set of managed objects, the result set may be too large to fit in a single response message. Therefore, enumerations consist of more than a single request/response pair. This example illustrates the communication flow of an enumeration operation.
Event Subscriptions	When subscribing for the delivery of events in the WMS System, there are three general message-flows that cover the different scenarios: pull subscriptions, push subscriptions, and publisher-initiated subscriptions. This example illustrates the communication flow of subscription and event delivery with these three scenarios.

6.1.1.5 Application Services

[MS-MQSO]: Message Queuing System Overview

Example	Description
Disconnected Data Entry	An application requiring exactly-once delivery guarantee to transfer business messages asynchronously between machines.
Web Order Entry	An application requiring independent scaling out of the front-end application servers and the back-end processing servers.
Modify a Public Queue	A management application requiring the ability to administer Message Queuing operations locally and remotely.
Creating and Monitoring a Remote Private Queue	A management application requiring the ability to configure and monitor queues locally and remotely.
Branch Office Order Processing	An application requiring asynchronous and efficient message transfer functionality in a complex network topology.
Business-to-Business Messaging Across	An application requiring message communication between two or

Example	Description
Firewall	more business units across organizational network boundaries.
Server Farm	An application requiring scaled-out processing of messages that are asynchronously and reliably transferred from multiple application machines to a central intermediary server.
Stock Ticker	An application requiring broadcasting of messages from a central machine to multiple recipient machines.
Business-to-Business Messaging Across Heterogeneous Systems	An application requiring asynchronous message communication between two or more business units when at least one business unit has a different message queuing system other than the Message Queuing (MSMQ) System.

[MS-TPSO]: Transaction Processing Services System Overview

Example	Description
Basic Transaction Life Cycle	This example shows how transaction life-cycle roles interact with each other in the life time of the transaction.
Two-Phase Commit	This example is the continuation of the example Basic Transaction Life Cycle and shows how transaction life-cycle roles interact with each other during the Two-Phase Commit protocol.
Transaction Is Aborted	This example is the continuation of the example Basic Transaction Life Cycle and shows how transaction life-cycle roles interact with each other during the Two-Phase Commit protocol where one of the participants aborts the transaction.
Connection to Resource Manager Breaks Down	This example is the continuation of the example Basic Transaction Life Cycle and shows how transaction life-cycle roles interact with each other during the Two-Phase Commit protocol where connection to a resource manager breaks down, and how the resource manager drives recovery afterwards.
Transaction Manager Recovers after a Connection Failure	This example is the continuation of the example Basic Transaction Life Cycle and shows how transaction life-cycle roles interact with each other during the Two-Phase Commit protocol where connection to a subordinate transaction manager breaks down, and how the transaction manager drives recovery afterwards.
Distributed Transaction Coordination with External Components - Distributed Transaction Coordination with External Components Precursory Message Exchanges	This example shows the message exchanges that occur at the initialization time of the participants.
Distributed Transaction Coordination with External Components - Application-Driven Transactional Message Exchanges	This example shows the message exchanges that occur when an application creates a transaction and other participants enlist in the transaction.
Distributed Transaction Coordination with External Components - Two-Phase Commit Transactional Message Exchanges	This example shows the message exchanges that occur when an application requests its Transaction Manager to drive the transaction to a conclusion by using the Two-Phase Commit protocol.

6.1.1.6 General Protocols

The following tables summarize the examples for the protocols that support collaboration, media services, protected content, remote desktop services (terminal services), and update services functionality.

[MS-CSSO] Collaboration Services System Overview

Example	Description
Collaboration Client Sending Registration Information to an ILS Server	The Collaboration Client does not automatically discover the ILS Server. The location of the ILS Server is entered manually by the user within the Collaboration Client application. This example illustrates Collaboration Client sending registration information to an ILS Server. The client utilizes several LDAP operations in order to accomplish the request
Collaboration Client Retrieving Registration Information	This example illustrates Collaboration Client retrieving registration information that other Collaboration Clients have placed on the ILS Server. The client will utilize several LDAP functions in order to accomplish the requests.

[MS-MGSO]: Multiplayer Games System Overview

Example	Description
Discovering, Joining, and Leaving a Local Area Network DirectPlay 8 Game Session	This scenario provides a conceptual overview of the process of discovering an existing DirectPlay 8 session on a Local Area Network.
Joining a DirectPlay 8 Host and Existing Peer With Network Address Translation Extensions	This scenario provides a conceptual overview of a nascent peer joining a previously established DirectPlay 8 game session with a host and existing peer.

[MS-MSSO]: Media Streaming Server System Overview

Example	Description
Encoder Push to Media Server	A key scenario is getting content from the server and onto the media server. One way the MSS System enables the encoder to get content onto the media server is through a push.
Media Server Pull from Encoder	In addition to the push model, the MSS System also supports a pull model for getting content from an encoder. The example illustrates the communication flow between the encoder and the server during a pull mode.
Multicast Playback	A key scenario for the MSS System is playing back multimedia content. The architectural details of this protocol are not expanded by the MSS System.
Unicast Playback	This example illustrates the communication flow between the Player Application and the server. The process for streaming multicast is broken down into four major areas: discovery, configuring the ports, stream selection, and playback.
Packet-Pair Bandwidth Estimation	This example illustrates packet-pair bandwidth estimation communication flow. To estimate bandwidth, the media server sends two or more consecutive packets of highly entropic data, and the client estimates the bandwidth by measuring the difference between

	the times that it receives the packets.
Advanced Fast Start/Fast Start	Fast Start and Advanced Fast Start work together to optimize the playback experience for the end user. This example shows the communication between a media player client, an origin server and distribution server taking into account the Advanced Fast Start and Fast Start headers.
Logging	Logging is a process that allows the media player to submit information and statistics to a media server or a Web server. This example shows the logging communication between a Media Player Application, the Media Player Client Role, a media server Server Role, and a Web Server.
Integrating DRM	Digital Rights Management (DRM) integration happens entirely outside the MSS System. This example shows the logging communication between a Media Player Application, the Encoder Application, the DRM Packager, the DRM Server, the MSS Server, and the MSS Client.

[MS-RMSO]: Rights Management Services System Overview

Example	Description
Protecting Content Using Offline Publishing	This example illustrates the typical steps that can be completed for a user and computer that has not used RMS in the past to bootstrap the client, acquire a Client Licensor Certificate (CLC), acquire templates and, finally, publish the content offline.
Protecting Content Using Online Publishing	In online publishing the client computer is not required to be activated and the client user is not required to be bootstrapped. This example describes a typical scenario using online publishing, assuming rights policy templates are used in the publishing process.
Consuming Protected Content	This example illustrates the typical steps performed to consume content, assuming the computer and user have not used RMS in the past.

[MS-TSSO]: Terminal Services System Overview

Example	Description
Connecting from an RDP Client to a TS Server	This example illustrates the RDP Client state model for a basic connection scenario. In this scenario, an RDP client connects to a TS Server in an intranet environment where no gateway is used.
Connecting from an RDP Client to a TS Server through a TS Gateway	This example illustrates communication messages within the Terminal Services System for establishing a connection between an RDP Client and TS Server using a TS Gateway.
Establishing a Dynamic Virtual Channel for Plug and Play Device Redirection	One common scenario for using a dynamic virtual channel involves redirecting data from a Plug and Play device that is connected to an RDP Client after a user session has been established. This example illustrates the sequence establishing the dynamic virtual channel.
Redirecting Clipboard Data	A static virtual channel is used to redirect Clipboard data. This example illustrates how a Clipboard channel is initialized and subsequently used to transfer data.
Disconnection Sequence	This example illustrates two ways an RDP Client may leave a connection with a TS Server: The RDP Client is disconnected from a TS Server due to network or other reasons, and the user of the RDP

Example	Description
	Client logs off from the TS Server.

[MS-WSUSO]: Windows Server Update Services System Overview

Example	Description
Update Synchronization to DSS	This example describes the scenario where a downstream server (DSS) is configured to synchronize with an upstream server (USS) and the synchronization is triggered using the server management tool or triggered on a schedule from the downstream server.
Initial Approval Sync to Replica DSS	When the DSS is configured as a replica of the USS, it synchronizes target groups and update approvals also from the USS in addition to updates. This example describes the message exchanges required to accomplish this synchronization when a DSS communicates with a USS.
Initial Update Sync to Update Client	The goal of this example is for a particular Update Client to synchronize update metadata and deployments from a particular Update Server for the first time. In this case, the Update Client has no cached data from previous synchronizations with the Update Server.
Differential Update Sync to Update Client	The goal of this example is for a particular Update Client to synchronize update metadata and deployments from a particular Update Server after having already synchronized at a previous point in time. In this case, the Update Client has cached data from previous synchronizations with the Update Server, which it uses to optimize the synchronization.
Rollup of Reporting Data to USS	In this example, the goal of the scenario is for the DSS to send update installation and applicability information about the clients and descendent DSSs to the USS.
Update Client is Pointed to a New Update Server	In this example, the goal of the scenario is for a particular Update Client to synchronize update metadata and deployments from a different Update Server than it used for its previous synchronization.

6.2 Communication Details

In the Communication Details section of many System Overview documents, the system does not define additional communication details beyond those described in the specifications of the protocols supported by the system. The following documents fall into this category; [MS-MSSO], [MS-PSSO], [MS-RMSO], [MS-TPSO], [MS-TSSO], [MS-WMSO], and [MS-WSUSO].

For the rest of the System Overview documents, the Communication Details are as follows.

- [MS-ADSO]

  Some of the protocols in the Active Directory System can operate on more than one transport. The constraints are as follows:

  - While the Active Directory System supports both TCP and UDP transports for LDAP versions 2 and 3, TCP is the preferred transport.
  - For SAMR, all opnums are exposed over both the Server Message Block (SMB) and TCP transports except that SamrValidatePassword (opnum 67) is only exposed via the TCP transport. Clients MUST use the TCP transport when performing this operation.

- For LSAT, clients MUST choose which transport to use (either SMB or TCP) based on the opnum of the request that they are sending to the server.

The system does not define additional communication details beyond those described in the specifications of the protocols supported by the system.

- [MS-CASO]

  The system components communicate though the shared ADM elements that are listed in the Abstract Data Model section in [MS-CASO] (section 6.1).

  All messages and their processing rules are documented in the protocol specifications for the protocols that this system implements as specified in [MS-CASO] section 2.2.

- [MS-FSSO]

  In protocol layering, each protocol may be transported by a number of lower-layer protocols, and the lower-layer protocol connection must be established before the given protocol connection can be established. This system does not define additional communication details beyond those described in the specifications of the protocols supported by the system.

- [MS-GPSO]

  The two communication process flows of interest are the interactions between the Group Policy Client and the Group Policy server and the interaction between the Administrative tool and the Group Policy Server as follows:

  - Protocol communication between a Group Policy Client and Group Policy Server ([MS-GPSO] section 6.2.1): Communication between the GP Client and the GP Server uses core protocols to transport GP specific information. Protocol communication to and from the Administrative Tool and Group Policy Server ([MS-GPSO] section 6.2.2): Group Policy is managed with an administrative tool that uses the same protocols and, in several instances the same protocol sequence methods that the GP Client itself uses.

- [MS-MGSO]

  The DirectPlay System is capable of using multiple underlying transports, such as UDP/TCP, IPX, or serial connections. However, firewall and Network Address Translation issues only apply to UDP/TCP (IP-based) transports, therefore the [MC-DPLNAT] extensions are not relevant to other transports.

- [MS-MQSO]

  The communications within the Message Queuing System and between the system and external entities are described in [MS-MQSO] sections 5.4.1 and 5.4.2 respectively. The system does not define additional communication details beyond those described in the specifications of the protocols supported by the system.

6.3 Transport Requirements

In this section of many System Overview documents, the system does not define a transport or define any transport requirements beyond those described in the specifications of the member protocols. The following documents fall into this category; [MS-FSSO], [MS-MSSO], [MS-RMSO], [MS-TPSO], [MS-WMSO], and [MS-WSUSO].

For the rest of the System Overview documents, the Transport Requirements are as follows.

- [MS-ADSO]

Clients communicate with the directory service using RPC-based, block-structured, and SOAP protocols. The Web Services protocols use SOAP 1.2 [SOAP1.2-1/2003] as the message format over the "net.tcp" transport defined in [MC-NMF]. WS-Addressing is bound to SOAP 1.2 as described in [WSASB].

- [MS-CASO]

  The system specific transport requirements are defined in [MS-WCCE] section 2.1, [MS-ICPR] section 2.1, and [MS-CSRA] section 2.1.

- [MS-GPSO]

  The GP Client uses SMB and LDAP for transmitting Group Policy settings, and for transmitting instructions between the GP Client and the GP Server. The GP Client uses DC Locator/DNS for finding the GP Server; Kerberos and SPNEGO for authenticating the computer for computer policy application; and SPNEGO for user policy application.

- [MS-MGSO]

  In the DirectPlay 4 System protocols, normal [MC-DPL4CS] operations and application-specified payloads are transported using [MC-DPL4CS] message types. When the application has enabled voice functionality, it uses an API provided by the voice protocol implementation to invoke its functionality.

- [MS-MQSO]

  The following transports are used within the Message Queuing system as well as with external entities:

  - Transports Used Within the System ([MS-MQSO] section 6.3.1): The MQQB protocol is used to transfer a message from an outgoing queue to a destination queue with a range of delivery assurances. This block protocol uses either TCP/IP or SPX/IPX as the underlying data transport. The SRMP protocol is also used to transfer messages from an outgoing queue to a destination queue with a range of delivery assurances.

  - Transports Used Between the Systems ([MS-MQSO] section 6.3.2): Applications communicate with the queue manager over several RPC-based and DCOM-based protocols. Applications and queue managers interact with the Directory Service through LDAP.

- [MS-PSSO]

  The Print Services System uses transports as described in the component protocol documentation. There is no system wide transport security. Each component protocol specifies its own transport security.

- [MS-TSSO]

  RDP Client to TS Server communication requires Transmission Control Protocol (TCP) transport over either an IPV4 or IPV6 network.

6.4 Timers

In this section of many System Overview documents, the system has no timer events or does not define any transport requirements beyond those described in the specifications of the member protocols. These documents fall into this category; [MS-ADSO], [MS-CASO], [MS-MSSO], [MS-PSSO], [MS-RMSO], [MS-TPSO], and [MS-WMSO]. In the case of [MS-TPSO] an implementation of the system is required to provide the timers described in the specifications of the protocols supported by the system.

For the rest of the System Overview documents, the System Timers are described below.

- [MS-FSSO]

  The timers for this system are as follows:

  - SMB Session Auto-Disconnect Timer ([MS-FSSO] section 6.4.1)
  - SMB Oplock Break Response Wait Timer ([MS-FSSO] section 6.4.2)
  - SMB Scavenger Thread Timer ([MS-FSSO] section 6.4.3)

- [MS-GPSO]

  For Group Policy the client SHOULD have the Periodic Refresh timer. This timer SHOULD be triggered periodically to check for updated policy for the computer or each user interactively logged on to the computer. The frequency of this timer is implementation specific as described in [MS-GPSO].

  [MS-MGSO]

  The DirectPlay System timers are confined to affecting only the associated Member Protocol in which they are documented. However, because of the largely vertical relationship among Member Protocols, a timer may cause a lower layer protocol to transition to a state that appears as an external non-timer event to a higher layer.

- [MS-MQSO]

  The timers of the Message Queuing System are described in the specifications of the protocols supported by the system. The following timers are those that are important to the state of the overall system, and the Message Queuing System wants to maintain them:

  - Directory Service Synchronization Timers ([MS-MQSO] section 6.4.1): The synchronization timers in this section are enabled only if the Message Queuing System is operating in Directory-Integrated with Routing mode.
  - Message Timers ([MS-MQSO] section 6.4.2): The message timers are described in [MS-MQDMPR] sections 3.1.2.4 and 3.1.2.5, and the associated timer events are described in [MC-MQAC] sections 3.1.6.1.1 and 3.1.6.1.2.
  - Security Timers ([MS-MQSO] section 6.4.3): This section discusses the following security timers:
    - Certificate Data Cache Validity Timer: This timer regulates the amount of time that a queue manager operating in Directory-Integrated mode waits before invalidating the internal certificate data cache.
    - Queue Manager Public Key Cache Validity Timer: This timer regulates the amount of time that the queue manager waits before invalidating the cache.

- [MS-TSSO]

  The Session Disconnect timer is the primary timer that is used by the TS Server to disconnect an idle session. The TS Server disconnects the session on a timeout event by initiating a protocol disconnect event.

- [MS-WSUSO]

  An update server SHOULD use a timer to trigger periodic synchronization with its upstream server (USS). The frequency of the timer is implementation-specific. Similarly, an update client SHOULD use a timer to trigger periodic reporting to its update server. The frequency of the timer is implementation-specific. The periodic reporting of the update client to its update server impacts the update server's USS.<3>

6.5 Non-Timer Events

In this section of many System Overview documents, the system has no non-timer events or does not define any non-timer events beyond those described in the specifications of the member protocols. The following documents fall into this category: [MS-FSSO], [MS-MQSO], [MS-MSSO], [MS-RMSO], [MS-WMSO], and [MS-WSUSO].

For the rest of the System Overview documents, the System Timers are described as follows.

- [MS-ADSO]

  There is one non-timer event, Host Name Change, in the Active Directory System (beyond those non-timer events specified in the underlying protocol documents).

- [MS-CASO]

  The system-specific events are defined in [MS-WCCE] section 3.2.1.4, [MS-ICPR] section 3.2.4, and [MS-CSRA] section 3.1.4.

  A system implementing the enterprise CA mode SHOULD register for the asynchronous change notifications to the certificate template objects in Active Directory. See [MS-WCCE] section 3.2.2.5.

- [MS-GPSO]

  On the client, policy application in computer policy mode SHOULD be invoked at the time that the computer boots or connects to a new network and MAY be invoked at other times. Policy application in user policy mode are be invoked at the time a user logs in or connects to a new network and MAY be invoked at other times.

- [MS-MGSO]

  Because of the largely vertical nature of the DirectPlay System's relationships among Member Protocols, many external events apply equally to each protocol. Since each DirectPlay System Member Protocol encapsulates the state of those below it, the state of the system as a whole is effectively the same as the state of the highest level protocol.

- [MS-PSSO]

  In the event of a system shutdown the Print Services System MUST close all open handles to remote objects and unregister endpoints. When a local Plug and Play event occurs; when new printer hardware is detected on a local port of a computer acting as Print Server, it automatically locates a matching driver for the detected printer and installs a Print Queue. The created Print Queue can optionally be set up as a shared Print Queue (per policy setting).

- [MS-TPSO]

  This section describes one local event, when a connection is disconnected ([MS-TPSO] section 6.5.1.1), and one external event ([MS-TPSO] section 6.5.1.2.1), system recovery.

- [MS-TSSO]

The Terminal Services System depends on platform facilities for creating a remote session for the remote user. As such, there are several error events possible during this operation arising from several subsystems such as authentication, authorization, group policy, and licensing.

6.6 Initialization and Reinitialization Procedures

In this section of many System Overview documents, there are no system initialization or reinitialization requirements beyond what is described in the individual protocol Technical Documents. The following documents fall into this category; [MS-FSSO], [MS-MQSO], [MS-MSSO], [MS-RMSO], [MS-TPSO], and [MS-WMSO].

For the rest of the System Overview documents, the System Timers are described as follows.

- [MS-ADSO]

  The Active Directory System does not contain a requirement to reinitialize the protocols, nor does it contain specific procedures for reinitializing the protocols.

- [MS-CASO]

  The initialization processes are defined in [MS-WCCE] sections 3.2.1.3 and 3.2.2.2, [MS-ICPR] section 3.2.3, and [MS-CSRA] section 3.2.3.

- [MS-GPSO]

  During initialization, the Group Policy client SHOULD register for computer boot and user logon notification. As part of re-initialization, the Group Policy client SHOULD recreate the operational state pertaining to the machine and every logged-on user.

- [MS-MGSO]

  Initialization and reinitialization of the DirectPlay System is controlled by the application. An application should be designed to use either DirectPlay 4 or DirectPlay 8. The protocols to be used should be initialized together, at the same time, since higher layers depend on lower ones and all share similar lifetimes.

- [MS-PSSO]

  Initialization occurs at system startup, which will start the print spooler service. Reinitialization occurs up to two times only upon abnormal termination of the print spooler service.

- [MS-TSSO]

  The initialization of the RDP Client, TS Server, and TS Gateway are described in this section of [MS-TSSO]. Other components of the Terminal Services System are initialized as described in the component protocol documentation. Virtual channels, including dynamic virtual channels (DVC), are initialized as a part of the connection sequence between the RDP Client and the TS Server.

- [MS-WSUSO]

  Initialization of the system entails the following:

  - Update servers generate a globally unique identifier to identify themselves to other update servers.
  - Update clients generate a globally unique identifier to identify themselves to update servers.
  - Update servers are initialized with the location (for example, DNS name and port) of their USS.

- Update clients are initialized with the location (for example, DNS name and port) of their update server.

An update server or update client can be individually reinitialized without reinitializing the entire system.

6.7 Status and Error Returns

In this section of many System Overview documents, the system does not define any error handling requirements beyond those described in the specifications of the member protocols. The following documents fall into this category: [MS-FSSO], [MS-GPSO], [MS-MGSO], [MS-MQSO], [MS-MSSO], [MS-PSSO], [MS-RMSO], [MS-TPSO], [MS-TSSO], [MS-WMSO], and [MS-WSUSO].

For the rest of the System Overview documents, the Transport Requirements are as follows.

- [MS-ADSO]

  The Active Directory System does not define any error handling requirements beyond those described in the Technical Documents of the protocols supported by the system and does not have a dedicated configuration or administration protocol. It is configured using the same protocols as otherwise used by applications for interacting with the system.

  This document also adds a section on System Management Details ([MS-ADSO] section 6.8).

- [MS-CASO]

  The error handling requirements are defined in [MS-WCCE] section 2.2.4 and [MS-CSRA] section 2.2.5.

7 Appendix A: Product Behavior

The information in this specification is applicable to the following product versions:

- Microsoft Windows® 95 operating system
- Microsoft Windows® 98 operating system
- Windows® Me operating system
- Microsoft Windows NT® operating system
- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows® XP operating system Service Pack 1 (SP1)
- Windows® XP operating system Service Pack 2 (SP2)
- Windows® XP operating system Service Pack 3 (SP3)
- Windows Server® 2003 operating system
- Windows Server® 2003 R2 operating system
- Windows Server® 2003 operating system with Service Pack 1 (SP1)
- Windows Server® 2003 operating system with Service Pack 2 (SP2)
- Windows Vista® operating system
- Windows Vista® operating system with Service Pack 1 (SP1)
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. The new behavior also applies to subsequent service packs of the product unless otherwise specified.

Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Supported in Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008.

<2> Windows NT 4.0 and earlier domains allowed the use of a period (.) at the end of a user logon name as long as the user logon name did not consist solely of period characters. Windows Server 2003 domains do not allow the use of a period or multiple periods at the end of a user logon name.

<3> Domain join is supported in Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008.

<4> Active Directory is supported in Windows 2000, Windows Server 2003 and Windows Server 2008.

<5> Supported in Windows Server 2003 and Windows Server 2008.

<6> The list of applicable servers follows: Windows 2000, Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008.

<7> Active Directory domain controllers (DCs) that do not accept changes are a new feature for Windows Server 2008. Windows 2000 servers and Windows Server 2003 servers cannot be configured this way. Windows NT 4.0 domains have one DC that accepts changes, and all other DCs are read-only.

<8> That is, Netlogon also creates a general purpose channel for authentication. It is not specific to any protocol and is available only to components involved in authentication.

<9> Windows 98 through Windows Vista (RTM version inclusive) use algorithms that are compliant with the FIPS 140-2 standard. Windows Vista SP1 uses a new algorithm which is compliant with the NIST SP 800-90 standard. The random number generation in all versions of Windows prior to Windows Vista SP1 is consistent with FIPS 186-2 Change Notice 1.

<10> Windows uses the cryptographic PRNG from the Cryptographic API (CAPI) and the Cryptographic API Next Generation (CNG) for generation of Version 4 GUIDs.

<11> For reasons of increased privacy protection for our customers, Microsoft systems beginning with Windows 2000 prefer to generate version 4 GUIDs in which the 122 bits of nonformat information are random. Although only a small minority of version 4 GUIDs require cryptographic randomness, the random bits for all version 4 GUIDs built in Windows are obtained via the Windows CryptGenRandom cryptographic API or the equivalent, the same source that is used for generation of cryptographic keys. This source is FIPS 140-certified in various versions of Windows, as documented at [MSFIP140CryptCerts].

<12> [RFC1510] (the original Kerberos RFC) was made obsolete and was replaced by [RFC4120].

<13> Prior to Windows 2000, the only supported impersonation levels were identify and impersonate. In Windows 2000, delegate-level impersonation is supported.

<14> This level is supported beginning with Windows 2000.

8 Appendix B: Scenarios

This section includes two supplemental scenarios with use cases for the PowerShell and BranchCache protocols..

8.1 Power Shell

Until recently, management of Windows Protocols Systems required several tools; one for monitoring system activity (done through the Windows Management protocols, using CIM object models for managed objects) and a separate tool for actually taking action on a remote system (such as Terminal Services). The new PowerShell capability allows a system manager to treat a remote system exactly the same as the system on their desktop, by providing a complete remote shell capability. More details on functionality can be found in [MS-PSRP].

This remote shell capability has the ability to configure, start, monitor, and stop ANY possible process supported by the software on the remote system. It also supports standard shell features such as the setting of environment variables, and a complete shell scripting language. This provides a true remote management capability, allowing a system manager complete control over the entire distributed environment.

8.1.1 Use Cases

8.1.1.1 Use Case Diagram

The following diagram illustrates the use case described in section 7.1.1.2.

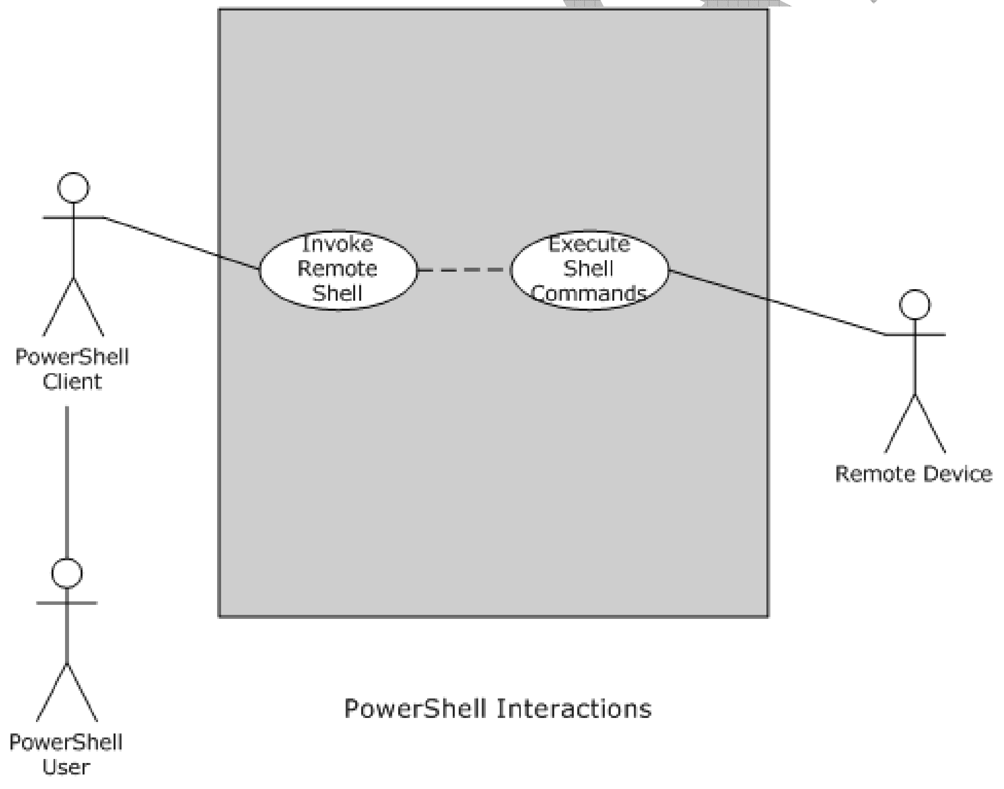


Figure 20: PowerShell use case diagram

8.1.1.2 Invoke a Remote Shell – Windows PowerShell Client

Goal: To use PowerShell to issue shell commands to a remote device. The remote device will execute the shell commands, accept input to the shell commands from the local device, and return any shell output to the local device.

Context of Use: This use case typically occurs when a PowerShell user wishes to run commands on a remote device.

Direct Actor: The direct actor in this use case is the PowerShell client.

Primary Actor: The primary actor in this use case is the PowerShell user.

Supporting Actors: The supporting actors in this use case are the PowerShell protocol and the remote device.

Stakeholders and Interests:

- The PowerShell user, for executing commands on the remote system.
- The remote system, for executing the requested commands.
- The PowerShell client, for transmitting the commands to the remote system.
- The Authentication System, as specified in [MS-AUTHSO].

Preconditions:

- The remote system must be reachable over the network from the local system.
- The PowerShell user must have the required rights to execute the desired commands on the remote system.
- The remote system must have the capability of carrying out the issued commands.

Minimal Guarantee: None

Success Guarantee: PowerShell guarantees that the requested remote commands were executed, and that the output from those commands is delivered to the local computer.

Trigger: This use case is triggered by a request from the PowerShell user.

Main Success Scenario:

1. The PowerShell client establishes a remote shell session with the remote device.
2. The client accepts the desired commands from the user, and transmits them to the remote device for execution.
3. The remote device executes the commands.
4. The PowerShell protocol returns the output from the commands to the PowerShell client.

Extensions: None.

8.1.2 Architectural Details

This section provides examples illustrating the flow of messages between a client and server using the PowerShell Remote Shell execution capability. The examples are:

- Execute a command that runs successfully on a remote computer
- Execute a command that returns an error when executed on a remote computer

8.1.2.1 Communication Flow

The following diagram shows the communication flow used by the examples described in sections 7.1.2.2 and 7.1.2.3.

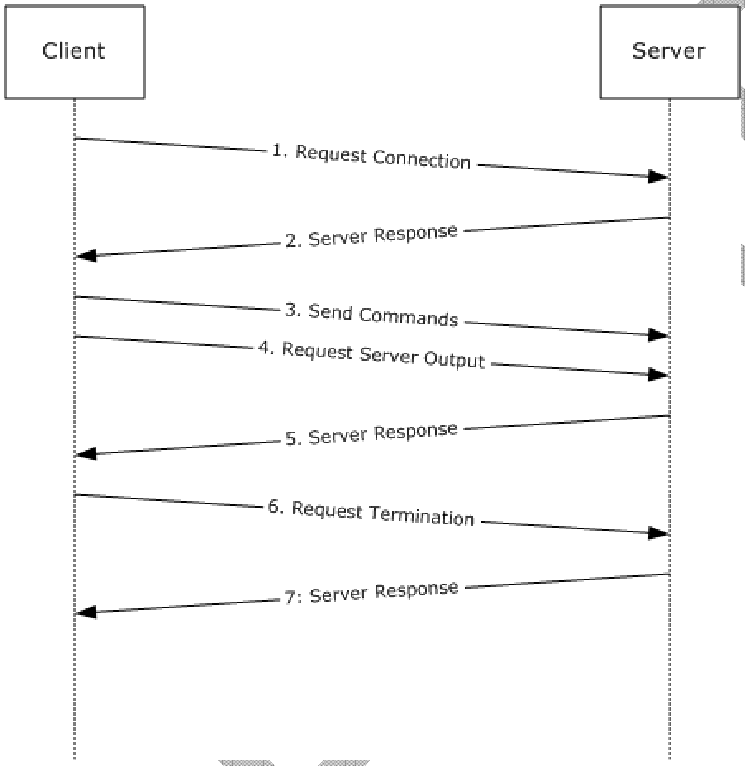


Figure 21: Communication flow for executing a command

8.1.2.1.1 Request Connection

The client role connects to the server role and receives a response. See [MS-PSRP] Section 3.1.4.1 for details.

8.1.2.1.2 Server Response

The server role responds to the request for connection. See [MS-PSRP] Section 3.2.5.4.1 and 3.2.5.4.3 for details.

8.1.2.1.3 Send Commands for execution

On successful connection to the target machine, the client role can send commands for execution to server role. As long as the connection is active, the server role executes the command and sends the results back to the client role. See [MS-PSRP] section 3.1.4.3 for details.

8.1.2.1.4 Request Server Output for Execution Result

As PowerShell defaults to NOT receiving the server output from the executed commands, the client must explicitly request that it receives the output from the commands. See [MS-PSRP] section 3.2.5.4.2.

8.1.2.1.5 Server Response

Server returns output from the executed commands. See [MS-PSRP] section 3.2.5.1.2 for details.

8.1.2.1.6 Request Terminate Connection

To close the connection, the client role sends a request for connection termination to the server role. See [MS-PSRP] section 3.1.4.2 for details.

8.1.2.1.7 Server Response

Server returns response to the request. See [MS-PSRP] section 3.1.5.2.6 for details.

8.1.2.2 Example: Successful Command Execution

The following example assumes that the target computer has two applications running: Calc.exe (Calculator) and Notepad.exe (Notepad).

1. User creates remote connection to the target computer and stores the connection state in a variable.

   $s = New-PSSession –ComputerName localhost

2. User invokes the command on target computer, using the connection variable, to find the calculator and notepad processes state.

   Invoke-Command -Session $s -ScriptBlock {Get-Process calc,notepad}

   The output of the previous command execution should look similar to the following:

   ```
   Handles  NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName  PSComputerName
   -------    ------      -----    -----    -----    ------    --    -----------          --------------
        85       7          5576   10608    69     0.23   7168   calc                    localhost
       76       4          1168     6704    72     0.08   5108   notepad              localhost
   ```

3. User invokes the command on the target machine, using the connection variable, to stop the calculator and notepad processes.

Invoke-Command -Session $s -ScriptBlock {Stop-Process –Name calc,notepad}

4. User closes the remote connection to the target machine using the connection variable. This step maps to [MS-PSRP] section 1.1.1.3.

   Remove-PSSession $s

8.1.2.3 Example: Command Execution Returns Error

The following example assumes that the target computer does not have the Calc.exe (Calculator) and Notepad.exe (Notepad) processes running.

1. User creates remote connection to the target machine and stores the connection state in a variable.

   $s = New-PSSession –ComputerName localhost

2. User invokes the command on target computer, using the connection variable, to stop the calculator and notepad.

   Invoke-Command -Session $s -ScriptBlock {Stop-Process calc,notepad}

   The output of the previous command execution should look similar to the following:

```
Cannot find a process with the name "calc". Verify the process name and call the cmdlet
again.

        + CategoryInfo: ObjectNotFound: (calc:String) [Stop-Process],
        ProcessCommandException

        + FullyQualifiedErrorId:
        NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.StopProcessComm
        and

Cannot find a process with the name "notepad". Verify the process name and call the
cmdlet again.

        + CategoryInfo: ObjectNotFound: (notepad:String) [Stop-Process],
        ProcessCommandException

        + FullyQualifiedErrorId:
        NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.StopProcessComm
        and
```

3. User closes the remote connection to the target computer using the connection variable.

   Remove-PSSession $s

8.1.3 Communication Details

The system does not define additional communication details beyond those specified in the underlying member protocol documents.

8.1.4 Transport Requirements

The system does not define a transport beyond those specified in the underlying member protocol documents. Note that [MS-PSRP] uses [MS-WSMV] as an underlying transport protocol.

8.1.5 Timers

There are no timer events beyond those specified in the member protocol documents.

8.1.6 Non-Timer Events

The system does not define any non-timer events beyond those specified in the underlying member protocol documents.

8.1.7 Initialization and Re-initialization Procedures

The system does not define any initialization or re-initialization procedures beyond those specified in the underlying member protocol documents.

8.1.8 Status and Error Returns

The system does not define any status and error returns beyond those specified in the underlying member protocol documents.

8.2 Branch Caching

8.2.1 Architectural Details

The Windows 7 and Windows Server 2008 R2 operating systems introduce the feature BranchCache. This enables content from file and web servers on a wide area network (WAN) to be cached on computers at a local branch office. Cached content can be distributed across peer client computers (Distributed Cache mode) or centrally hosted on a server (Hosted Cache mode).

When BranchCache is enabled, a copy of the content that is retrieved from the server using (http/https or smb2) is cached within the branch office. If another client in the branch office requests the same content, that client can download it directly from a cached location in the local branch network without needing to retrieve the content by using the Wide Area Network (WAN).

Depending on where the cache is located, BranchCache can operate in one of two modes:

- Hosted Cache
- Distributed Cache

8.2.1.1 Distributed Cache Mode

BranchCache Distributed mode involves multiple peers operating in cooperation. The peers use the Discovery Protocol [MS-PCCRD], an implementation of the Web Services Dynamic Discovery (WS-Discovery) protocol and the Retrieval Protocol described in [MS-PCCRR] to allow the client-role peer to find and retrieve content blocks of a target segment from one or more server-role peers. The framework accepts requests from the higher-layer applications on the client-role peer to retrieve the whole or parts of a content item, which may span multiple segments. For each target segment, the client-role peer utilizes the Discovery Protocol to find server-role peers that have (the whole or parts) of the target segment. The client-role peer then initiates Retrieval Protocol exchanges to each server-role peer to query the block ranges each server-role peer has, and downloads the blocks.

The Discovery Protocol and the Retrieval Protocol both operate on or within a single segment (see [MS-PCCRC] for specification of segments and blocks). The operations specified in this section allow a client-role peer to find and retrieve blocks (parts or all) of a single target segment. This process is referred to as a Segment Retrieval Session. If the content spans multiple segments, then the framework can use multiple Segment Retrieval Sessions to retrieve all the content's segment and reassemble them into the complete content item.

Distributed Cache mode requires clients to support [MS-PCCRC] for content identification, [MS-PCCRR] for content query and retrieval, and [MS-PCCRD] for per server service discovery. The clients keep a copy of the content and make it available to other authorized clients that request the same data. This eliminates the need to have a server in the branch office. However, unlike Hosted Cache mode, this configuration works across a single subnet only (that is, the content has to be retrieved once per subnet in the branch office by using the WAN).

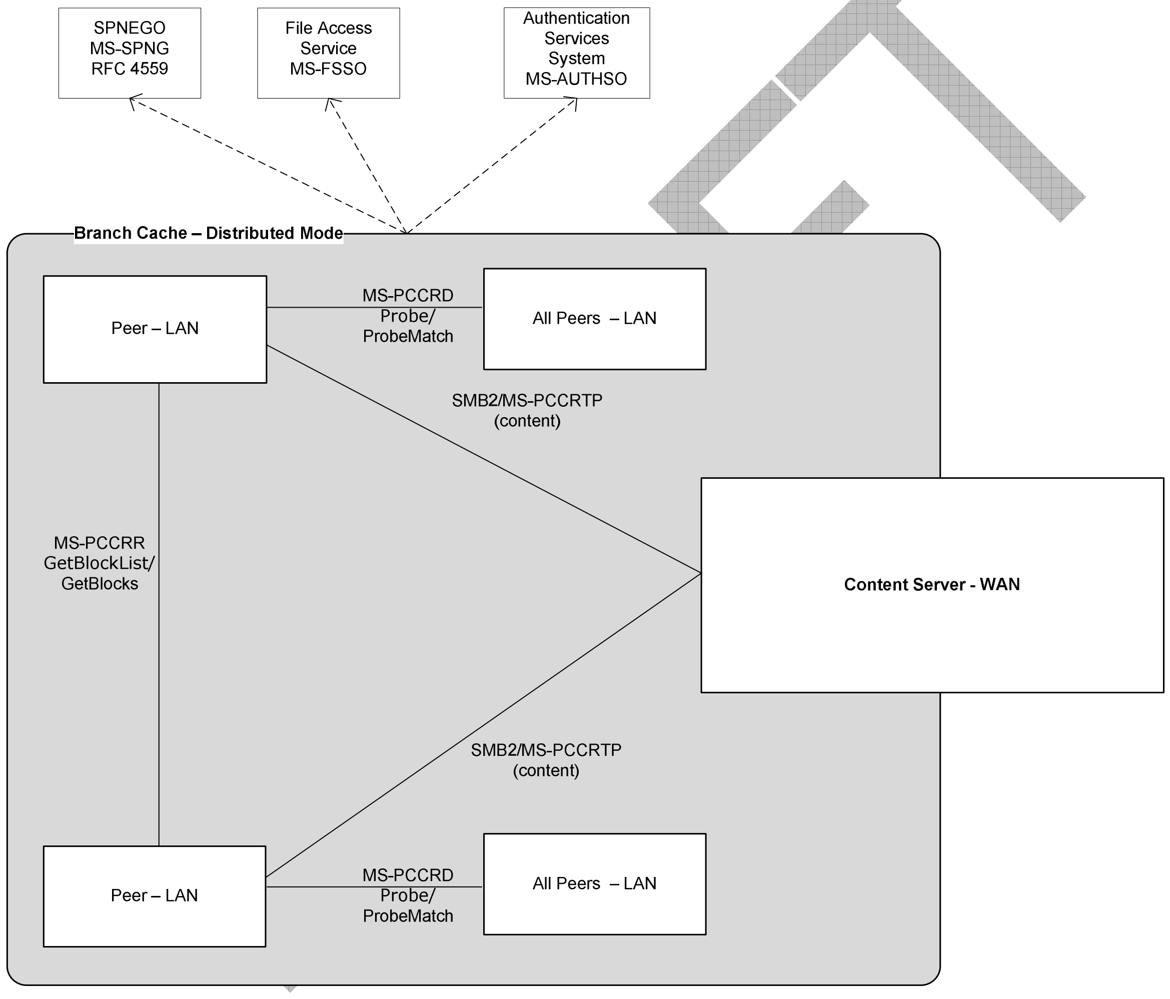


Figure 22: BranchCache - Distributed Mode

At a more detailed level, the Distributed Cache mode uses the following process to cache and retrieve data:

1. A client connects to the content server and requests a file, exactly as it would if it were to retrieve the file without using cache.

2. The content server authenticates [MS-AUTHSO] and authorizes the client [MS-DISO] as appropriate, (note that BranchCache does not require domain services although they may be used) and the server returns an identifier that the client uses to search for the file on the local network. If it is the first time any client has attempted to retrieve the file, it will not be already cached on the local network. Therefore, the client retrieves the file directly from the content server and caches it. See [MS-SMB2] or [HTTP].
3. A second client requests the same file from the content server. The content server authenticates and authorizes the user in exactly the same manner it would if cache were not being used. If successful, it returns content metadata over the same channel that data would normally have been sent. See [MS-SMB2] or [HTTP].
4. The second client sends a request on the local network for the required file by using the Web Services Discovery (WS-Discovery) multicast protocol. See [MS-PCCRD] for details of content discovery.
5. The client that previously cached the file sends the file to the requesting client. The data is encrypted by using a key that is derived from the hashes sent by the content server as part of the content metadata. See [MS-PCCRC].
6. The receiving client decrypts the data, computes the hashes on the blocks received from the first client, and ensures that it is identical to the block hashes provided as part of the content metadata by the content server. This ensures that the content has not been modified.

8.2.1.2 Hosted Cache Mode

The clients in the Peer Content Caching and Retrieval Framework can also work in a hosted cache mode. The difference here is that in hosted cache mode, the clients do not use the Discovery Protocol to discover peer servers to retrieve content blocks. Instead, the clients are configured with the address of the hosted cache, and query the hosted cache directly for the block ranges of the requested segments. The hosted cache, in turn, is offered content by peers (clients) and retrieves the offered content blocks directly from the peer that offered them. Therefore, in the following description, the roles of "client" and "server" are used instead of "peer" and "hosted cache".

The Hosted Cache server does not maintain any state other than receiving and processing the requests as defined in the Retrieval Protocol details.

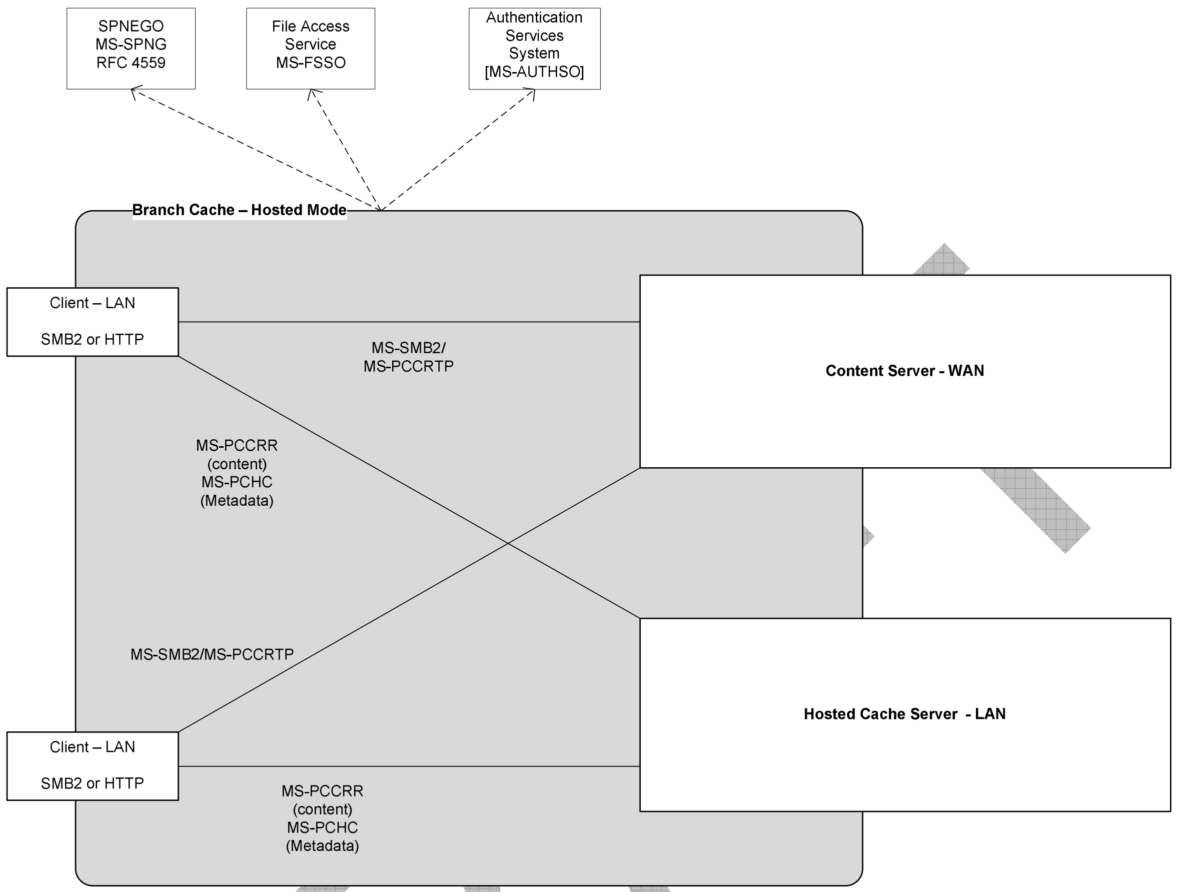


Figure 23: Hosted Cache Mode

The Hosted Cache mode requires a server that supports the protocols defined in: [MS-PCCRR], [MS-PCHC], and [MS-PCCRC] as a host.<1>

Clients that can utilize the Hosted Cache support the protocols defined in: [MS-PCCRR], [MS-PCHC], [MS-PCCRC], [MS-PCCRTP] and [MS-SMB2].<2>

Clients of Hosted Cache mode are configured with the fully qualified domain name of the host computer of a cache so that they can retrieve content from the Hosted Cache, when available. If the content is not available in the Hosted Cache, it is retrieved from the Content Server by using the WAN and then offered to the Hosted Cache making it available to other clients. That is always the case the first time content is retrieved as there will be no cached until at least one client has retrieved the full file.

At a detailed level, Hosted Cache mode uses the following process to cache and retrieve data:

1. The client connects to the Content Server and requests a file exactly as it would if it were to retrieve the file without using cache (Using [MS-SMB2] or [MS-PCCRTP]).

2. The Content Server authenticates (see [MS-AUTHSO]) and authorizes (see [MS-DISO]) the client exactly as it would without using Cache. If successful, it returns content metadata over the same channel that data would normally have been sent. See [MS-PCCRR].

3. The client uses hashes in the metadata to search for the file in the Hosted Cache server as described in [MS-PCRR] if this is the first time any client has retrieved the file; it will not be already cached on the local network. Therefore, the client retrieves the file directly from the Content Server.

4. The client establishes a Secure Sockets Layer (SSL) connection with the Hosted Cache server, and it offers the content identifiers over this encrypted channel. See [MS-PCHC].

5. The Hosted Cache server connects to the client and retrieves the set of blocks that it does not have cached. See [MS-PCCRR].

6. A second or subsequent client requests the same file from the Content Server. Again, the Content Server authorizes the user and returns content identifiers (metadata).

7. The client uses these content identifiers to request the data from the Hosted Cache server. The Hosted Cache server encrypts the data and returns it to the client. (The data is encrypted by using a key that is derived from the hashes sent by the content server as part of the content metadata.) See [MS-PCCRC].

8. The client decrypts the data, computes the hashes on the blocks received from the Hosted Cache, and ensures that it is identical to the block hashes that the content server provided as part of the content metadata. This ensures that the content has not been modified.

8.2.1.3 Content Metadata

The mechanism for reducing bandwidth usage is to send metadata about the content (known as content metadata) to clients, which retrieve the content from within the branch rather than over the WAN. This reduces the WAN bandwidth because the content metadata is usually significantly smaller than the actual content. Prior to sending content metadata, the server authorizes the client.

It is important that the content server sends the content metadata to each client to ensure that the client always receives hashes for the most up-to-date content.

The content is broken into blocks. For each block, a hash is computed (known as the block hash). A hash is also computed on a collection of blocks (known as the segment hash). Content metadata is primarily composed of block hashes and segment hashes. The hash algorithm that is used is normally SHA-256, but it is configurable to SHA-384 and SHA-512 (see [MS-PCCRC] for the details of the hashing process and the structures involved).

Segment hashes provide a unit of discovery. This helps reduce the total number of lookups performed for a given content (compared to looking up every block). Block hashes are a unit of download. When a client needs to retrieve data from the Hosted Cache or another client, it downloads the content in units of blocks to ensure that the data can quickly return to the application.

The minimum size of content that BranchCache would cache is 64 KB. When content is less than 64 KB, data rather than metadata is directly retrieved from the content server by using the WAN. See [MS-PCCRR] for the specification of the Caching and Retrieval, and [MS-PCCRC] for the specification of the content information format.

8.2.2 Communication Details

BranchCache supports the SMB 2 and HTTP 1.1 protocols. Applications do not need to directly communicate with BranchCache although they can if they need to.<4> However, applications accessing SMB2 and HTTP interfaces in the Windows 7 and Windows Server 2008 R2 operating systems transparently benefit from BranchCache when it is enabled.

8.2.2.1 Protocols Used by the Peer Content Caching and Retrieval Framework

- Peer Content Caching and Retrieval: Content Identification [MS-PCCRC]

  Content Identification is a binary data structure that contains all necessary information to allow peers to discover other peers with requested content, ensure the confidentiality of content sent between peers and verify the integrity of downloaded content blocks.

- Peer Content Caching and Retrieval Discovery Protocol Specification [MS-PCCRD]

  The Peer Content Caching and Retrieval Discovery Protocol is based on the Web Services Dynamic Discovery (WS-Discovery) protocol, which uses multicast to discover and locate services over the network. There are two modes of operations in WS-D: client-initiated Probes and service-initiated announcements; both are sent through IP multicast to a predefined group. The Peer Content Caching and Retrieval Discovery Protocol uses the client-initiated Probes to query peer nodes for content. The peer role server responds with Probe-Match messages.

- Peer Content Caching and Retrieval: Retrieval Protocol Specification [MS-PCCRR]

  The Retrieval Protocol defines two request/response exchanges between a client and a server on top of an HTTP [RFC2616] transport – to query the availability of specific content, and to retrieve specific content. The protocol assumes the client knows both the specific content it is looking for, and the server it will contact.

- Peer Content Caching and Retrieval: Hosted Cache Protocol Specification [MS-PCHC]

  The Peer Content Caching and Retrieval: Hosted Cache Protocol provides a mechanism for clients to inform the hosted cache about segment availability. There are two primary roles:

  - Client: The client informs the hosted cache that it has segments it can offer.

  - Hosted cache: The hosted cache gets the range of block hashes associated with the segment being offered, and then downloads the blocks within the segment that it actually needs.

- Peer Content Caching and Retrieval: Hypertext Transfer Protocol (HTTP) HTTP Client Extensions [MS-PCCRTP]

  HTTP/1.1 is a client/server-based protocol. The purpose of the PeerDist content encoding is to enable peer content caching and retrieval in HTTP/1.1 and HTTPS Using the PeerDist content encoding, the client can participate in the peer content caching and retrieval process. Upon seeing the PeerDist encoding support from a client, the HTTP/1.1 server can send a PeerDist encoded response.

  Server Message Block 2 [MS-SMB2] Server Message Block 2.1 has enhancements for the detection a utilization of BranchCache enabled shares.

8.2.3 Security

This section describes the security of data during transportation and at rest (in the client cache or Hosted Cache).

A typical sequence of events is as follows:

1. A client requests data from the content server, and indicates that it is capable of understanding BranchCache.

2. The content server authenticates and authorizes the client in exactly the same way it would if BranchCache were not being use. That is, authentication and authorization of a client to access data are independent of BranchCache.

3. The content server recognizes that the client can utilize BranchCache (note that there are different mechanisms for SMB2 and HTTP), and checks to make sure that the stored metadata is up to date with the content.

4. The content server then sends the metadata on the same channel that data normally would have been sent. If an SSL connection were established between the client and the server, then the hashes are sent back over this encrypted SSL connection.

5. The client that is requesting content obtains the metadata and uses it to look up availability in the branch.

6. The client establishes a connection with the caching computer (a Hosted Cache server when Hosted Cache mode is used, or a peer caching computer when Distributed Cache mode is used), and requests the blocks it requires.

7. The caching computer encrypts the blocks with an encryption key that is derived from the content metadata (using AES 128 by default) and sends it to the client. The details of the encryption process for AES 128 can be found in [FIPS197].

8. The client decrypts the data by using the same encryption key that the caching computer. The client and the caching computer compute the same encryption key because they derive it from the same content metadata, which is sent by the content server.

9. After the client decrypts the data, it validates that the data. To do this, the client computes the block hashes on the blocks received, and then compares them to the block hashes received in the content metadata from the server. If the hashes do not match, the client discards the data.

The data in the cache is accessible. The data is stored in the clear in the Distributed Cache and the Hosted Cache, which is similar to other caches and data on the system (such as the IE cache, the SMB offline files cache, and file system).

8.2.4 Transport Requirements

[MS-PCCRTP] transport is Hypertext Transfer Protocol (HTTP) over Transport Layer Security (TLS) (HTTPS) [RFC2818].

[MS-PCCRD] transport is [WS-Discovery].

[MS-PCHC] transport is Hypertext Transfer Protocol (HTTP) over Transport Layer Security (TLS) (HTTPS) [RFC2818].

[MS-PCCRR] transport is Hypertext Transfer Protocol (HTTP) [RFC 2616].

8.2.5 Timers

[MS-PCCRTP] - None

[MS-PCCRD] - Two timers are associated with the Discovery Protocol operations, backoff and request. See section 3.1.2 [MS-PCCRD]

[MS-PCHC] - None

[MS-PCCRR] – Request timer, Upload timer see section 3.1.2 [MS-PCCRR]

8.2.6 Non-Timer Events

[MS-PCCRTP] -None

[MS-PCCRD] - Receive Probe, Receive Probe-Match; see section 3.1.5 [MS-PCCRD]

[MS-PCHC] - None

[MS-PCCRR] - GetBlockList Initiation in section 3.1.4.1

[MS-PCCRR] - GetBlocks Initiation in section 3.1.4.2

8.2.7 Initialization and Re-initialization Procedures

[MS-PCCRTP] - The HTTP Client is initialized as per section 3.1.4 of [MS-CCRTP]

[MS-PCCRD] - Probe and Probe-Match messages are initialized as per section 3.1.3 of [MS-PCCRD]

[MS-PCHC] - The hosted cache is initialized as per section 3.1.3 of [MS-PCHC]

[MS-PCCRR] - The server side is initialized as per section 2.1.1 of [MS-PCCRR]

8.2.8 Status and Error Returns

[MS-PCCRR] Block Download Status – For each block in the requested block ranges of the segment, a flag is stored to indicate the download status (idle, downloading, downloaded, or error)

[MS-PCCRD]) - Message processing and error handling refer to [WS-Discovery]

[MS-PCHC] - HTTP Status Code 401 Response Received - The client MUST resend the request, indicating to the transport that SPNEGO-based HTTP authentication should be performed

8.2.8.1 Abstract Data Model

[MS-PCCRTP] - None

[MS-PCCRD] – The following ADM items are described in detail in section 3.1.1:

- Client Peer **-** HoHoDk List, Outstanding Probe List
- Server Peer - Available Segment List

[MS-PCHC] - The following ADM items are described in detail in section 3.1.1:

- Content information for the offered segment, Block cache:
  - [MS-PCCRR]

The following ADM items are maintained by [MS-PCCRR] and described in detail in section 3.2.1:

- Content cache, Server Information Lists, Discovery Frequency Counter, Discovered Server Counter, Download Initiated Flag, Block Download Status

8.3 Detail Use Cases for Scenario

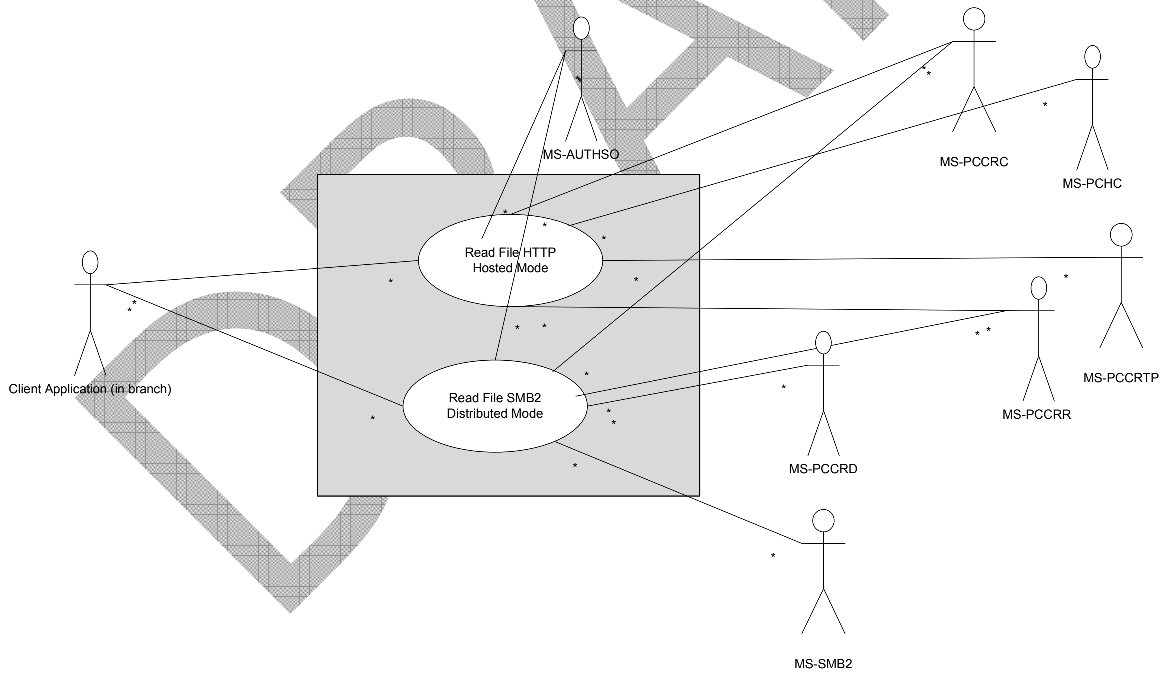

MS-AUTHSO
MS-PCCRC
MS-PCHC
Read File HTTP Hosted Mode
Client Application (in branch)
MS-PCCRTP
MS-PCCRR
Read File SMB2 Distributed Mode
MS-PCCRD
MS-SMB2

Figure 24: BranchCache Use Cases

8.3.1 Read a file using HTTP with Hosted BranchCache

Goal - Read a file from a content server website with BranchCache support enabled using HTTP

Context of use - User has located a file on a content server (a web server in this case) and wants to read that file and obtain a handle to that file. The file is located on a website with BranchCache Support enabled. The file is not initially on the local area network and may need to be retrieved across a wide area network.

Minimal Guarantees - No action is taken that affects other files exposed on the Content Server as a result of this operation.

Success Guarantee - User obtains a handle to the file.

Trigger - User action in Application.

8.3.1.1 Stakeholders and Interests Summary

The stakeholders and their associated interests for BranchCache Hosted Cache Mode – HTTP are as follows:

HTTP Application Client – (aka web browser) The HTTP Application Client's interest is to correctly interpret, execute and display the results of the commands issued by the User. The HTTP Application Client is external to the File Services System and BranchCache.

User - the User's interest is to use the HTTP protocols provided by the HTTP Client Application to access files on a Content Server.

Content Server's – The Content Server's interest in this Use Case is to provide and maintain a secure and consistent File Access Service see [MS-FSSO] and provide content metadata

The Hosted Cache Servers interest is to cache data and to receive metadata from clients regarding the availability of data segments and then download the blocks within the segment that it actually requires and to supply blocks of data to clients when requested

8.3.1.2 Supporting Actors and Task Interests Summary

MS-AUTHSO - Authentication Services provides authentication. - If the hosted cache is configured to require client authentication, both the client and the hosted cache are required to support SPNEGO-based HTTP authentication ([RFC4559] and [MS-SPNG]) within the HTTPS transport.

MS-PCHC's interest is in providing a mechanism for clients to inform the hosted cache about segment availability and to provide a mechanism for the hosted cache to download the blocks within the segment that it actually needs.

MS-PCCRR's interest is in providing two mechanisms, one for querying the Hosted Cache for the availability of certain content, and the other for retrieving content from a server

The Client Application's interest in this Use Case is to provide and maintain a consistent access mechanism to the HTTP Application Client.

8.3.1.3 Main Success Scenario

1. The User requests the Application to return a handle to an existing file.
2. The Application establishes a TCP connection to the Content Server for [HTTP]+ [MS-PCCRTP]
3. The Content Server authenticates the User through the mechanisms of [MS-DISO] if required.
4. The Application performs a GET request as specified in RFC2616 decorated with [MS-PCCRTP] extensions.
5. The Content Server checks the authorization of the User to perform the action [MS-AUTHSO] if required.
6. The Content Server retrieves metadata (block hashes, segment hashes, and private segment key) for the data [MS-PCCRR]
7. The Content Server sends metadata on same application channel to client [HTTP]
8. The Client computes a segment discovery key [MS-PCCRC]
9. The Client requests the segment [MS-PCCRR] from the Hosted Cache
10. Client receives the data from the Hosted Cache and decrypts the data [MS-PCCRR]
11. Client Validates block data against the block hash.
12. The Application supplies a file handle to the HTTP Application Client.

8.3.1.4 System Assumptions and Preconditions

The following preconditions must be satisfied for BranchCache – Hosted Mode to operate successfully:

Preconditions - The Client and Hosted Cache Server (Hosted Cache mode) computers within the branch are configured to use BranchCache. The Client is configured with the address of the server maintaining the hosted cache. A website with a file has been created on the Content Server with BranchCache support enabled. The user has located the URL to the file on the Content Server. A prior client within the branch has retrieved the data enabling some or all of the content to be stored in the hosted cache.

System Availability - BranchCache must be installed and enabled on all the computers involved.

Domain Configuration: In a domain configuration, Client and Servers have access to directory services provided by the domain. (Note Domain configuration is not a requirement)

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

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

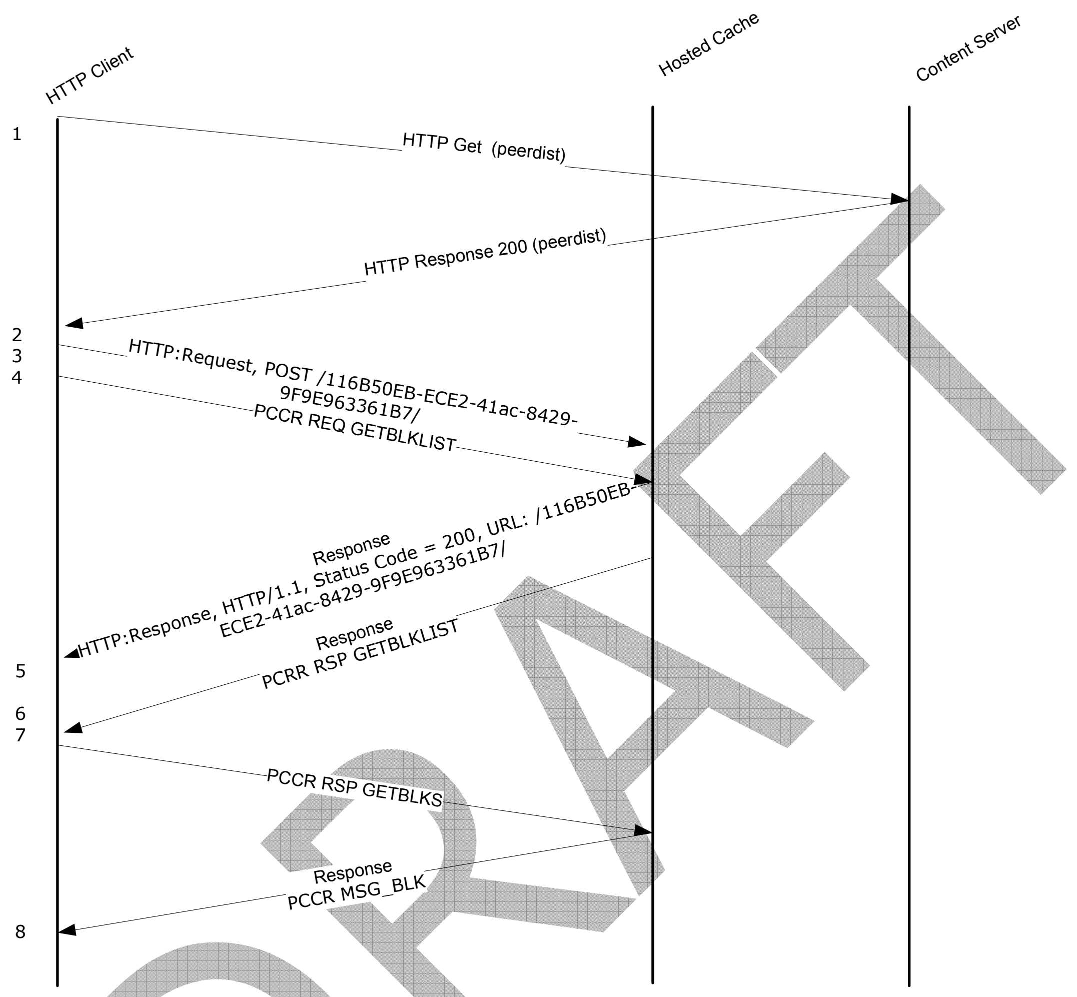
8.3.1.5 HTTP GET

The sequence described in this example details how the Application uses the HTTP Client to open a file on the content server.

8.3.1.5.1 Sequence of Events

1. The Client Application sends an HTTP GET request to the content server to gain access to the file on the content server. The client indicates in the http request header that it is peerdist enabled by listing peerdist as an accepted encoding [MS-PCCRTP]
2. In the response, the content server indicates that the content is encoded using the PeerDist content encoding, it also includes the content length of the entity-body the content-length indicates the length of the metadata.
3. The client peer initiates the transport by sending an HTTP POST request to the root path of {116B50EB-ECE2-41ac-8429-9F9E963361B7} on the Hosted Cache Server
4. The hosted cache server responds with a HTTP status code of 200 (OK) for the URL /116B50EB-ECE2-41ac-8429-9F9E963361B7
5. The client peer sends a [MS-PCCRR] REQUEST-MESSAGE (MSGGETBLKLIST) (carried as a HTTP POST) to the hosted cache server requesting the block IDs of the blocks within the target segment that the client-role peer is interested in.
6. The hosted cache server respond with a [MS-PCCRR]RESPONSE-MESSAGE (MSGGETBLKLIST) indicating the blocks currently available for download from the hosted cache serve.
7. The Client side http sends a number of [MS-PCCRR]REQUEST-MESSAGE (MSG_GETBLKS) to the hosted cache server
8. The hosted cache server responds with [MS-PCCRR]RESPONSE-MESSAGE(MSG_BLK)

Steps 7 and 8 are repeated until the required blocks are transferred.



HTTP Client
Hosted Cache
Content Server
1
HTTP Get (peerdist)
HTTP Response 200 (peerdist)
2
3
4
HTTP:Request, POST /116B50EB-ECE2-41ac-8429-9F9E963361B7/
PCCR REQ GETBLKLIST
Response
HTTP:Response, HTTP/1.1, Status Code = 200, URL: /116B50EB-ECE2-41ac-8429-9F9E963361B7/
Response
PCRR RSP GETBLKLIST
5
6
7
PCCR RSP GETBLKS
Response
PCCR MSG_BLK
8

Figure 25: Sequence Diagram - Read a File using HTTP with Hosted BranchCache

8.3.1.5.2 Assumptions, Scope, and Constraints

The use case assumes the cache is pre-populated with a copy of the file to be retrieved. No MS-PCHC traffic is expected due to this.

Expected protocols:

- HTTP from client to content server requesting the target file
- HTTP from client to hosted cache server
- MS-PCCRR transported by HTTP from client to hosted cache for obtaining cached content
- Data formats as described in MS-PCCRC
- Network Captures in promiscuous mode from the BranchCache server

8.3.2 Read a file using SMB2 with Distributed BranchCache

Goal - Read a file from a content server with BranchCache support enabled using SMB2

Context of use - User has located a file on a content server and wants to read that file and obtain a handle to that file. The file is located on a shared folder with BranchCache Support enabled. The file is not initially on the local area network and may need to be retrieved across a wide area network.

Minimal Guarantees - No action is taken that affects other files exposed on the Content Server as a result of this operation.

Success Guarantee - User obtains a handle to the file.

Trigger - User action in Application.

8.3.2.1 Stakeholders and Interests Summary

The stakeholders and their associated interests for Read a file using SMB2 with Distributed BranchCache are as follows:

SMB2 Application Client - The SMB2 Application Client's interest is to correctly interpret, execute and display the results of the commands issued by the User. The SMB2 Application Client is external to the File Services System and BranchCache.

User - the User's interest is to use the SMB2 protocol provided by the SMB2 Client Application to access files on a Content Server.

8.3.2.2 Supporting Actors and Task Interests Summary

MS-AUTHSO - Authentication Services provides authentication.

Content Server's – The Content Server's interest in this Use Case is to provide and maintain a secure and consistent File Service [MS-FSSO] and provide content metadata

MS-PCCRR's interest is in providing two mechanisms, one for querying the server for the availability of certain content, and the other for retrieving content from a server

The SMB2 Client Application's interest in this Use Case is to provide and maintain a consistent access mechanism to the required files on the Content Server.

8.3.2.3 Main Success Scenario

1. The User requests the Application to return a handle to an existing file.
2. The Application establishes a TCP connection to the Content Server for [MS-SMB2]
3. The Content Server authenticates the User through the mechanisms of [MS-DISO].
4. The Content Server checks the authorization of the User to perform the action. [MS-AUTHSO]
5. The Content Server retrieves metadata (block hashes, segment hashes, and private segment key) for the data [MS-PCCRR]
6. The Content Server sends metadata on same channel to client [MS-PCCRR]
7. The Client computes a segment discovery key [MS-PCCRC]
8. The Client broadcasts a Probe Message [MS-PCCRD]
9. A Peer with required content responds with a Probe Match message [MS-PCCRD]
10. The Client requests the segment from the peer [MS-PCCRR]
11. Client receives the data and decrypts the data [MS-PCCRR]
12. Client Validates block data against the block hash
13. The Application supplies a file handle to the User.

8.3.2.4 System Assumptions and Preconditions

The following preconditions must be satisfied for BranchCache – Distributed Mode to operate successfully:

Preconditions - The Client and Server (Hosted Cache mode) computers within the branch are configured to use BranchCache. A shared folder with a file has been created on the Content Server with BranchCache support enabled. The user has located the URL to the file on the Content Server. A prior client within the branch has retrieved the data enabling some or all of the content to be stored in a peer cache.

System Availability - BranchCache must be installed and enabled on all the computers involved.

Domain Configuration: In a domain configuration, Client and Servers have access to directory services provided by the domain. (Note Domain configuration is not a requirement)

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

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

8.3.2.5 SMB2 Read File

The sequence described in this example details how the SMB2 Application Client uses SMB2 to read a file located on the content server that may be subject to BranchCache. Explicit details of the reading a file without BranchCache are described in MS-FSSO. If caching is unavailable the fall back position will be as described in [MS-FSSO].

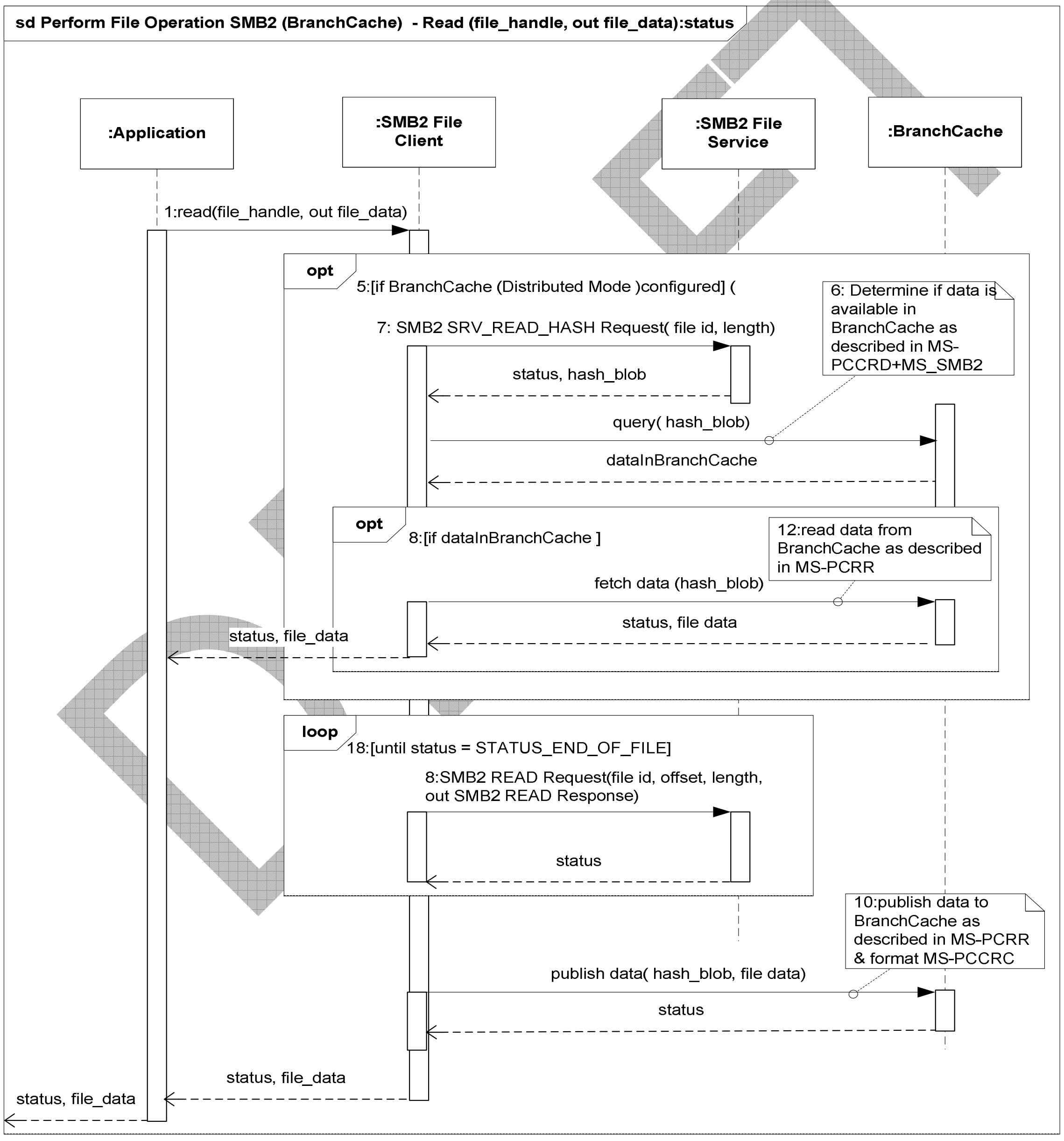

sd Perform File Operation SMB2 (BranchCache) - Read (file_handle, out file_data):status
:Application
:SMB2 File Client
:SMB2 File Service
:BranchCache
1:read(file_handle, out file_data)
opt
5:[if BranchCache (Distributed Mode)configured] (
6: Determine if data is available in BranchCache as described in MS-PCCRD+MS_SMB2
7: SMB2 SRV_READ_HASH Request(file id, length)
status, hash_blob
query(hash_blob)
dataInBranchCache
opt
8:[if dataInBranchCache]
12:read data from BranchCache as described in MS-PCRR
fetch data (hash_blob)
status, file data
status, file_data
loop
18:[until status = STATUS_END_OF_FILE]
8:SMB2 READ Request(file id, offset, length, out SMB2 READ Response)
status
10:publish data to BranchCache as described in MS-PCRR & format MS-PCCRC
publish data(hash_blob, file data)
status
status, file_data
status, file_data

Figure 26: SMB Read Operation with BranchCache

8.3.2.5.1 Sequence of Events

1. The SMB2 Client Application extracts the remote file server, the SMB2 File Share, and the target directory from the UNC path, unc_path, and invokes open() to direct the SMB2 File Client to gain access to the file and return a handle that the SMB2 Client Application can use in subsequent operations against the file. [MS-FSSO]

2. The SMB2 File Client examines internal private state to determine whether or not there is an existing session between the SMB2 File Client and the SMB2 File Service that can be use to satisfy the request from the SMB2 Application. If a session is required the SMB2 File Client proceeds to operation 3, otherwise it continues at operation 4. [MS-FSSO]

3. The SMB2 File Client attempts to establish a session using the sequence "SMB2 Session Setup" described in section [MS-FSSO] SMB Session Setup 6.1.5.23

4. The SMB2 File Client determines whether or not there is a valid tree connection for the SMB2 File Share between the SMB2 File Client and the SMB2 File Service. If there is no tree connection the SMB2 File Client proceeds to step 5, otherwise it continues to step 6. The process is further described in [MS-SMB2] section 3.2.4.2.4.

5. The SMB2 File Client attempts to establish a tree connection between the SMB2 File Client and the SMB2 File Service using an SMB2 TREE_CONNECT Request (see [MS-SMB2] section 2.2.9) using the previously obtained session identifier session_id and the UNC path to the share. The SMB2 File Service sends an SMB2 TREE_CONECT Response (see [MS-SMB2] section 2.2.10) with details of the tree connection, including an identifier tree_id the SMB2 File Client can use to identify subsequent requests using the tree connection. The SMB2 TREE_CONECT Response indicates in the returned ShareFlags field whether the share supports hash generation for BranchCache retrieval of data. The value SHI1005_FLAGS_ENABLE_HASH 0x00002000 indicates that BranchCache is enabled on the server share and that the server can respond to requests for metadata. This informs the client that it can make requests for metadata.

6. If BranchCache is enabled the client then attempts to obtain a lease that allows the client to cache read and writes to the data, see [MS-SMB2] Requesting a Lease on a File. A lease is obtained as part of the SMB2 CREATE request on accessing a file. That lease is requested as part of a CREATE with a RequestedOplockLevel of SMB2_OPLOCK_LEVEL_LEASE. It is only if this lease is granted by the server is it safe for the client to cache the data and therefore attempt to look up the data in the BranchCache.

7. If BranchCache is enabled on the server share, the SMB2 File Client performs a SRV_READ_HASH Request on the file and tries to obtain the hash blob for the target file. If the hash is out of date, an error is returned to the client; if no hash exists for the specified file, an error is returned to the client. <3>A server may choose to update the hash for either of these situations

8. On successfully obtaining a file hash, the client broadcasts a probe message for the required content [MS-PCCRD]

9. Any peers with the required content respond with a Probe-Match Message [MS-PCCRD]. The Probe-Match Message includes the HoHoDk which contains segment identifiers.

10. If no Probe-Match Message is received the full data will be retrieved from the content server. [MS-SMB2]

11. If the client receives a Probe-Match message it verifies that the response was sent by a peer on the local subnet, and checks the HoHoDks to verify that at least one is associated with a Probe message on the Outstanding Probe List. [MS-PCCRD]

12. The client then retrieves data by initiating the transport with the peer by sending an HTTP POST request to the root path of {116B50EB-ECE2-41ac-8429-9F9E963361B7} of the peer with the data [MS-PCCRR].

13. The peer responds with a HTTP status code of 200 (OK) for the URL /116B50EB-ECE2-41ac-8429-9F9E963361B7 [MS-PCCRR]

14. The client peer sends a [MS-PCCRR] REQUEST_MESSAGE (MSGGETBLKLIST) to the server-role peer cache requesting the block IDs of the blocks within the target segment that the client-role peer is interested in.

15. The server-role cache server respond with a [MS-PCCRR] RESPONSE_MESSAGE (MSGGETBLKLIST) indicating the blocks currently available for download from the hosted cache serve.

16. The Client side http sends a number of [MS-PCCRR] REQUEST_MESSAGE (GETBLKS) to the hosted cache server.

17. The hosted cache server responds with [MS-PCCRR] RESPONSE_MESSAGE(MSG_BLK).

Steps 12 to 16 are repeated until the required blocks are transferred.

18. If the operation was successful, the SMB2 File Client generates a handle, which is returned to the SMB2 Client Application for use when performing subsequent operations against the file.

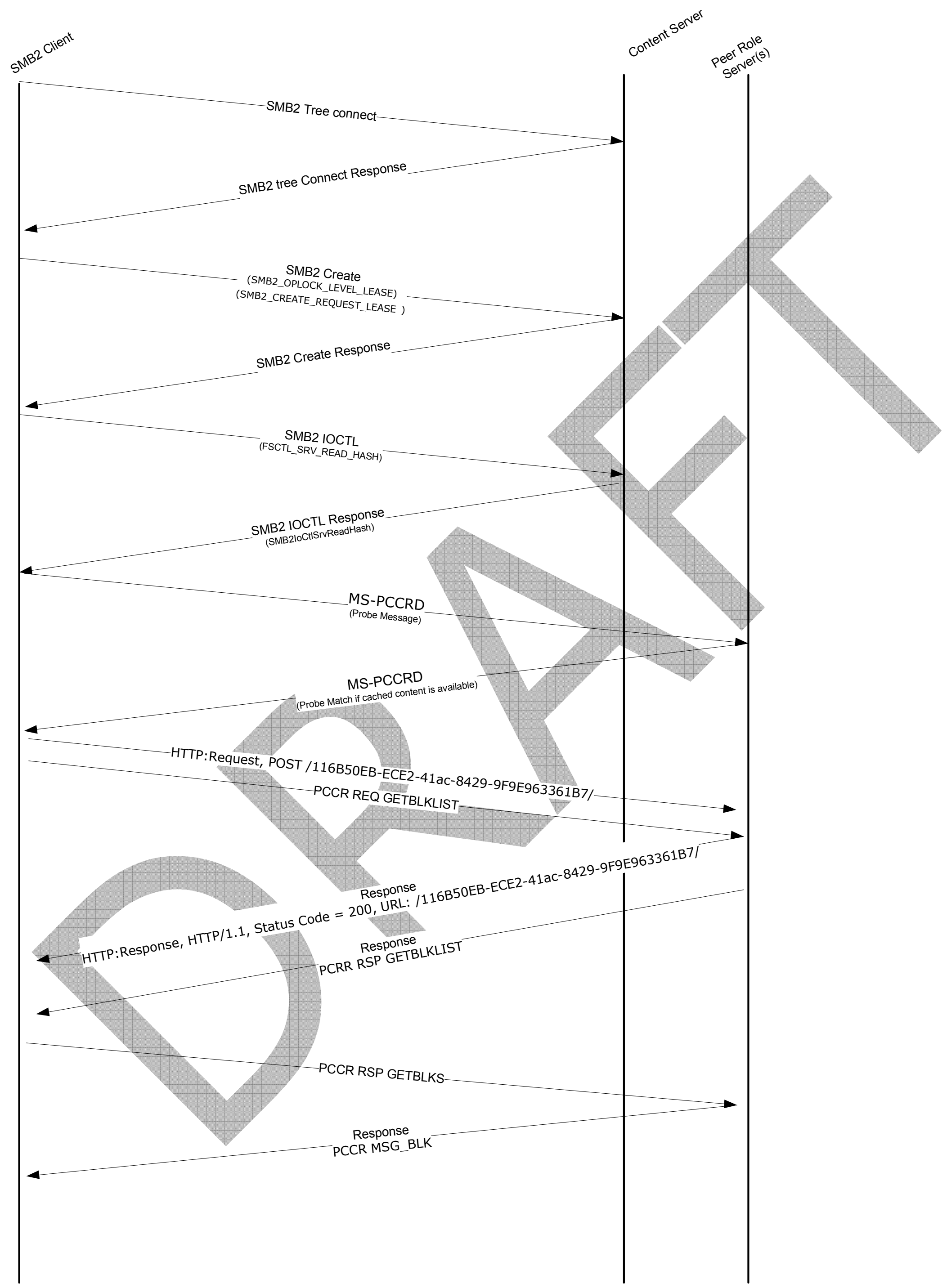


Figure 27: Diagram detail for read File SMB2 with distributed cache

8.3.2.5.2 Assumptions, Scope and Constraints

The use case assumes the cache is pre-populated with a copy of the file to be retrieved.

Expected protocols:

SMB2 as per [MS-SMB2] from client to content server requesting the target file

PCCRR as per [MS-PCCRR] from client peer to server peers for obtaining cached content

Data formats as described in [MS-PCCRC]

8.3.2.5.3 Environment

Client, Windows 7 Enterprise 32 bit, BranchCache Enabled – Distributed Mode

Content and BranchCache Server Windows 2008 R2 Enterprise (64 Bit only available), directory shared with target files and branch cache enabled on the share

8.3.2.5.4 References

[FIPS180-2] Federal Information Processing Standards Publication, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[FIPS197] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES)", November 2001, http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[WS-Discovery] Beatty, J., Kakivaya, G., Kemp D., et al., "Web Services Dynamic Discovery (WS-Discovery)", April 2005, http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf

8.3.2.5.5 Product Behavior Notes for BranchCache Scenario

<1> Windows Server 2008 R2 is the first server to support the server feature BranchCache – Hosted Cache Mode. The Hyper-V Core and Home Server SKU are not BranchCache capable. All other server Windows Server 2008 R2 SKUs can act as BranchCache clients and/or content servers

<2> Windows Server 2008 R2 and Windows 7 support the client feature BranchCache – Hosted Cache Mode and Distributed Mode. BranchCache is available with Windows 7 Enterprise and Ultimate.

<3> In Windows Server 2008 R2, if a client requests a hash for a file or no hash exists it will trigger a user-mode service to re-create/create the hash for the specified file.

<4> On Windows 7 and Windows 2008 R2, there are two ways for an application to use BranchCache: Directly by calling the BranchCache platform APIs, or indirectly by calling WinINET/WebIO, or SMBv2 APIs, which are instrumented to use BranchCache underneath, but transparent to the applications.

9 Appendix C: Protocols Affected by Group Policy

Implementations of the Windows Protocols System, as listed in the following table, can be affected by Group Policy. Some of these protocols have one or more properties that can be configured by Group Policy.

Protocol name	TD Short Name
Active Directory Web Services Custom Action Protocol Specification	[MS-ADCAP]
Authentication Protocol Domain Support Specification	[MS-APDS]
Background Intelligent Transfer Service (BITS) Peer-Caching Peer Authentication Protocol Specification	[MS-BPAU]
Corporate Error Reporting Version 1.0 Protocol Specification	[MS-CER]
Component Object Model Plus (COM+) Protocol Specification	[MS-COM]
CardSpace Token Acquisition Protocol Specification	[MS-CTAP]
Distributed Component Object Model (DCOM) Remote Protocol Specification	[MS-DCOM]
Distributed File System (DFS) Referral Protocol Specification	[MS-DFSC]
Distributed Link Tracking Workstation Protocol Specification	[MS-DLTW]
Digital Rights Management License Acquisition Data Structure	[MS-DRM]
Directory Replication Service (DRS) Remote Protocol Specification	[MS-DRSR]
Firewall and Advanced Security Protocol Specification	[MS-FASP]
File System Control Codes	[MS-FSCC]
File Server Resource Manager Protocol Specification	[MS-FSRM]
Health Certificate Enrollment Protocol Specification	[MS-HCEP]
Internet Information Services (IIS) IMSAdminBaseW Remote Protocol Specification	[MS-IMSA]
IP over HTTPS (IP-HTTPS) Tunneling Protocol Specification	[MS-IPHTTPS]
Internet Information Services (IIS) Inetinfo Remote Protocol Specification	[MS-IRP]
Kerberos Protocol Extensions	[MS-KILE]
Link Layer Topology Discovery (LLTD) Protocol Specification	[MS-LLTD]
Microsoft Media Server (MMS) Protocol Specification	[MS-MMSP]
Message Queuing (MSMQ) Directory Service Protocol Specification	[MS-MQDS]
Message Queuing (MSMQ) Queue Manager Remote Read Protocol Specification	[MS-MQRR]
Microsoft Web Browser Federated Sign-On Protocol Extensions	[MS-MWBE]
Microsoft Web Browser Federated Sign-On Protocol Specification	[MS-MWBF]
NT LAN Manager (NTLM) Authentication Protocol Specification	[MS-NLMP]
Protected Extensible Authentication Protocol (PEAP) Specification	[MS-PEAP]
Plug and Play Remote (PNPR) Protocol Specification	[MS-PNPR]
Peer Name Resolution Protocol (PNRP) Version 4.0 Specification	[MS-PNRP]
Remote Assistance Protocol Specification	[MS-RA]

Remote Administration Protocol Specification	[MS-RAP]
Remote Certificate Mapping Protocol Specification	[MS-RCMP]
Remote Desktop Protocol Basic Connectivity and Graphics Remoting Specification	[MS-RDPBCGR]
Remote Desktop Protocol Composited Remoting V2 Specification	[MS-RDPCR2]
Remote Desktop Protocol Audio Output Virtual Channel Extension	[MS-RDPEA]
Remote Desktop Protocol Graphics Device Interface (GDI) Acceleration Extensions	[MS-RDPEGDI]
Rights Management Services (RMS) Client-to-Server Protocol Specification	[MS-RMPR]
Remote Procedure Call Protocol Extensions	[MS-RPCE]
Remote Procedure Call over HTTP Protocol Specification	[MS-RPCH]
Remote Procedure Call Location Services Extensions	[MS-RPCL]
Print System Remote Protocol Specification	[MS-RPRN]
Routing and Remote Access Server (RRAS) Management Protocol Specification	[MS-RRASM]
Removable Storage Manager (RSM) Remote Protocol Specification	[MS-RSMP]
Remote Shutdown Protocol Specification	[MS-RSP]
Security Account Manager (SAM) Remote Protocol Specification (Client-to-Server)	[MS-SAMR]
Kerberos Protocol Extensions Service for User and Constrained Delegation Protocol Specification	[MS-SFU]
Server Message Block (SMB) Protocol Specification	[MS-SMB]
Server Message Block (SMB) Version 2 Protocol Specification	[MS-SMB2]
Secure Socket Tunnelling Protocol (SSTP) Specification	[MS-SSTP]
Transport Layer Security (TLS) Profile	[MS-TLSP]
Terminal Services Terminal Server Runtime Interface Protocol Specification	[MS-TSTS]
W32Time Remote Protocol Specification	[MS-W32T]
Windows Client Certificate Enrollment Protocol Specification	[MS-WCCE]
Web Distributed Authoring and Versioning (WebDAV) Protocol Client Extensions	[MS-WDV]
Web Distributed Authoring and Versioning (WebDAV) Protocol Microsoft Extensions	[MS-WDVME]
Workstation Service Remote Protocol Specification	[MS-WKST]
Web Point-and-Print Protocol Specification	[MS-WPRN]
Web Services Management Protocol Extensions for Windows Server 2003	[MS-WSMAN]
Web Services Management Protocol Extensions for Windows Vista	[MS-WSMV]
Windows System Resource Manager (WSRM) Protocol Specification	[MS-WSRM]
Windows Update Services Client-Server Protocol Specification	[MS-WUSP]

10 Appendix D: Standards Implemented in Windows

10.1 Standards Implemented

Windows implements the following standards:

6to4 Protocol: This protocol standard is an optional interim mechanism for IPv6 sites to communicate with each other over the IPv4 network without explicit tunnel setup. It also allows IPv6 sites to communicate with native IPv6 domains via relay routers across an IPv4 network. The protocol additionally provides, as a side effect, an interim globally unique IPv6 address prefix to any site with at least one globally unique IPv4 address.

1394 Serial Bus Protocol 2 (SPB2): This protocol standard is used to encapsulate small computer system interface (SCSI) requests over an IEEE (Institute of Electrical and Electronics Engineers) 1394 bus. It provides support for IEEE 1394 printers, CD-ROM, DVD, scanner, and storage devices.

Address Resolution Protocol (ARP): ARP is used to dynamically map an IP address of an onlink node to its 48 bit Ethernet address. ARP is also used to detect duplicate IP addresses within a network.

ANSI Terminal Emulation: ANSI describes the control characters (or escape sequences) transmitted over the wire from a computer to a dumb terminal, or terminal emulator, that control the positioning and color of characters on the terminal/emulator screen. The supported character set (defined originally in ANSI X3.64-1979, superseded by ECMA-48) is described in: http://nightmare.org/textfiles/programming/FORMATS/ansix364.txt.

Further information on the MSDN website at: http://msdn.microsoft.com/library/en-us/randz/protocol/hyperterminal_ansi_emulation.asp.

ANSI Terminal Emulation: Wide Character Extension (ANSIW): ANSIW is derived from ANSI (see ANSI Terminal Emulation protocol entry) and describes the control characters (or escape sequences) transmitted over the wire from a computer to a dumb terminal, or terminal emulator, that control the positioning and color of characters on the terminal/emulator screen. ANSIW supports wide (double byte ASCII characters) in addition to single byte ASCII characters. The supported character set (defined originally in ANSI X3.64-1979, superseded by ECMA-48) is described in: http://nightmare.org/textfiles/programming/FORMATS/ansix364.txt

Further information on the MSDN website at: http://msdn.microsoft.com/library/en-us/randz/protocol/hyperterminal_ansiw_character_set_options.asp.

ANSI X3T12: ANSI X3T12 (FDDI) is a protocol standard that enables the transfer of information and control between a pair of data link layer service access points on a FDDI Network. Windows includes third party drivers which implement the protocol for communication with their devices. These are in the form of drivers for Network Interface Cards. There are no online copies. The ANSI specification can be obtained from: ANSI standards (published FDDI standards) American National Standards Institute, 1430 Broadway, New York, NY 10018, USA Attention: Sales Dept.

AT Attachment 8 - ATA/ATAPI Command Set (ATA8-ACS): This standard specifies the AT Attachment command set between host systems and storage devices. It provides a common command set for systems manufacturers, system integrators, software suppliers, and suppliers of intelligent storage devices. It includes the PACKET feature set implemented by devices commonly known as ATAPI devices. This standard maintains a high degree of compatibility with the AT Attachment Interface with Packet Interface - 7 (ATA/ATAPI-7) volume 1, INCITS 397-2004, and while providing additional functions, is not intended to require changes to devices or software that comply with previous T13 standards.

Serial ATA Revision 2.6: This protocol standard defines a high-speed serialized ATA data link interface (specifying Phy, Link, Transport, and Application layers). The serialized interface uses the command set from the ATA/ATAPI-6 standard, augmented with Native Command Queuing commands optimized for the serialized interface. The serialized ATA interface is defined in a register-compatible manner with parallel ATA to enable backward compatibility with parallel ATA drivers.

AT command (ITU-T v.250) Protocol: The AT modem command set is used to enable an entity (for example an operating system) to talk to a modem device in a standard way. The AT command set, for example, specifies what string should be sent to the modem to tell the device to prepare for dialing a number, open a line, dial, negotiate baud rate (speed), etc. This is strictly for communication between a host and a device, in this case a modem.

ATAPI: ATAPI is a local hardware interface to communicate with storage devices (hard disks). ATAPI stands for Advanced Technology Attachment Packet Interface.

ATM User-Network Interface (UNI) Protocol: The Asynchronous Transfer Mode (ATM) User-Network Interface (UNI) protocol standard specifies the interfaces to be used between ATM user devices (e.g. a system running Windows) and ATM network equipment (e.g., an ATM switch) for the purposes of setting up end-to-end connections between ATM user devices. For information on ATM, see: http://www.atmforum.com/.

Atom Publishing Protocol: The Atom Publishing Protocol (AtomPub) standard is an application-level protocol for publishing and editing Web resources. The protocol is based on HTTP transfer of Atom-formatted representations. The Atom format is documented in the Atom Syndication Format. A number of Windows Live services follow this protocol such as Windows Live Photos.

Atom Syndication Format 1.0: Atom is an XML-based file format that describes lists of related information known as "feeds". Feeds are composed of a number of items, known as "entries", each with an extensible set of attached metadata. For example, each entry has a title. The primary use case that Atom addresses is the syndication of Web content such as weblogs and news headlines to Web sites as well as directly to user agents.

Basic Profile Version 1.0: The WS-I Basic Profile 1.0 consists of a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications which promote interoperability.

Basic Profile Version 1.1: The WS-I Basic Profile 1.1 consists of a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability.

Basic Security Profile: Core and SAML: This profile restricts usage of WS-Security and the Security Assertion Markup Language (SAML) standards in SOAP Web services to aid interoperability.

Bluetooth: The Bluetooth protocol standard is a low bandwidth serial bus protocol to connect wireless devices to computers.

Bluetooth Advanced Audio Distribution Profile (A2DP): The Bluetooth Advanced Audio Distribution Profile (A2DP) defines the protocols and procedures that realize distribution of audio content of high-quality in mono or stereo on asynchronous connection-oriented (ACL) channels. The A2DP profile relies on the transport service capability provided by the Audio/Video Distribution Transport Protocol (AVDTP). The A2DP requirements are expressed in terms of end-user services, and by defining the features and procedures that are required for interoperability between Bluetooth devices in the audio distribution usage model.

Bluetooth Audio/Video Remote Control Profile (AVRCP): This profile defines the requirements for Bluetooth™ devices necessary for the support of the Audio/Video Remote Control usage case. The requirements are expressed in terms of end-user services, and by defining the features and procedures that are required for interoperability between Bluetooth devices in the Audio/Video Remote Control usage case. This profile specifies the scope of the Audio/Video Control (AV/C) Digital Interface Command Set (AV/C command set, defined by the 1394 Trade Association) to be applied, and it realizes simple implementation and easy operability. This profile adopts the AV/C device model and command format for control messages, and those messages are transported by the Audio/Video Control Transport Protocol (AVCTP). In this profile, the controller translates the detected user action to the A/V control signal, and then transmits it to a remote Bluetooth device. The functions available for a conventional infrared remote controller can be realized in this profile. The remote control described in this profile is designed specific to A/V control. Other remote control solutions using Bluetooth wireless technology may be applied for general Bluetooth devices including A/V devices.

Bluetooth Hands-Free Profile (HFP): The Hands-Free Profile defines the minimum set of functions such that an audio gateway can be used in conjunction with a hands-free device (e.g. installed in the car or represented by a wearable device such as a headset), with a Bluetooth® Link providing a wireless means for both remote control of the audio gateway by the hands-free device and voice connections between the audio gateway and the hands-free device.

Bluetooth Hardcopy Cable Replacement Profile: The Bluetooth Hardcopy Cable Replacement Profile (HCRP) protocol standard is a protocol for printing to Bluetooth printers.

Bluetooth Headset Profile (HSP): The Bluetooth headset profile defines the protocols and procedures that shall be used by devices implementing the usage model called 'Ultimate Headset'. The most common examples of such devices are headsets, personal computers, and cellular phones. In this model Windows is acting as the audio gateway and communicates with the Bluetooth profile exposed by the Bluetooth device using the headset profile protocol.

Bluetooth Object Push Profile Version 1.0: Bluetooth Object Push Profile (OPP) provides wireless connections by enabling systems to perform object exchange (OBEX) functions. This feature is only client-client since Bluetooth support is not in the Windows Server family of operating systems (2000, 2003, or Vista Server).

Bluetooth Personal Area Network User (PANU) Profile: Bluetooth Personal Area Network (PAN) provides wireless connections by enabling links between mobile computers, mobile phones, portable handheld devices, and connectivity to the Internet. This is a complete implementation of the PANU (PAN User) profile role as specified in the Bluetooth PAN specification by the Bluetooth SIG.

Bluetooth Serial Port Profile - (Bluetooth RFCOMM): The RFCOMM protocol standard emulates the serial cable line settings and status of an RS-232 serial port and is used for providing serial data transfer. RFCOMM connects to the lower layers of the Bluetooth protocol stack through the L2CAP layer. By providing serial-port emulation, RFCOMM supports legacy serial-port applications while also supporting the OBEX protocol among others. RFCOMM is a subset of the ETSI TS 07.10 standard, along with some Bluetooth-specific adaptations.

Cascading Style Sheets (CSS), Level 1: This data structure specifies level 1 of the Cascading Style Sheet mechanism (CSS1). CSS1 is a simple style sheet mechanism that allows authors and readers to attach style (e.g. fonts, colors and spacing) to HTML documents. The CSS1 language is human readable and writable, and expresses style in common desktop publishing terminology.

Cascading Style Sheets (CSS) 2.1: Cascading Style Sheets, level 2 revision 1 (CSS 2.1) is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents (e.g., HTML documents and XML applications). By separating the presentation style of documents from the content of documents, CSS 2.1 simplifies Web authoring and site maintenance. This entry is intended to cover CSS 2.0 as well as CSS 2.1.

Character Generator Protocol: The Character Generator Protocol standard is a debugging and measurement tool. A character generator service sends data without regard to the input.

Common NNTP Extensions: Extensions, including authentication, for NNTP. NNTP Server does not ship in Windows Vista client or Windows Longhorn server.

Cross Domain Request (XDR) Protocol: This protocol standard allows a client side object, for example an AJAX object, to make requests to a server outside the domain where it was downloaded. The client side object will add a request header along with the request to a server from another domain. The server can return a response header with the response body to allow the response to be surfaced on a client side object. If the response header is missing, the client side object will discard the response as if an error occured. The request header and response header provides an explict opt-in mechanism for client and server to communicate cross domain. This is new for Windows 7, but will ship down-level as part of IE8 deployment.

Daytime Protocol standard: The Daytime Protocol standard is a debugging and measurement tool. A daytime service sends the current date and time as a character string without regard to the input.

Devices Profile for Web Services (DPWS): The Web services architecture includes a suite of specifications that define rich functions and that may be composed to meet varied service requirements. To promote both interoperability between resource-constrained Web service implementations and interoperability with more flexible client implementations, this profile identifies a core set of Web service specifications in the following areas:

- Sending secure messages to and from a Web service
- Dynamically discovering a Web service
- Describing a Web service
- Subscribing to, and receiving events from, a Web service

In each of these areas of scope, this profile defines minimal implementation requirements for compliant Web service implementations.

DCE 1.1: Remote Procedure Call: This protocol standard defines wire format, behavior, and portable APIs for the execution of cooperating components using a remote procedure call model. The Windows implementation is described under the entry: "Remote Procedure Call Protocol Extensions".

DHCP Options and BOOTP Vendor Extensions: The Dynamic Host Configuration Protocol (DHCP) standard provides a framework for passing configuration information to hosts on a TCP/IP network. Configuration parameters and other control information are carried in tagged data items that are stored in the 'options' field of the DHCP message. The data items themselves are also called "options."

Differentiated Services (DIFFSERV): IETF RFC 2474 defines the Differentiated Services (DS) field, which supersedes the IPv4 TOS field (RFC791) and the IPv6 Traffic Class field. The DS field can have a value from 0-63 (inclusive) for marking the priority of an IP datagram; this value is called a Differentiated Services Code Point (DSCP) value.

Digest Access Authentication: The Digest authentication mechanism [RFC2617] [RFC2831] performs authentication between a client and a server based on a user name and a password. Digest can authenticate the client to the server, and optionally the server to a client; the latter is termed mutual authentication. Digest was originally specified as a native authentication method for HTTP/1.1, in [RFC2616], to serve as an improvement on HTTP's Basic Authentication. Digest's popularity grew, and it was covered as a SASL [RFC2222] mechanism by the specification in [RFC2831]. Once made into a SASL mechanism, Digest became available for other protocols such as LDAP [RFC2251]. This entry includes both digest and basic authentication.

Digital Living Network Alliance (DLNA) Home Networked Device Interoperability Guidelines: The DLNA Networked Device Interoperability Guidelines provide vendors with the information needed to build interoperable networked platforms and devices for the digital home. They specify the necessary standards and technologies that enable products to be built for networked entertainment-centric usages, allowing seamless sharing of music, photos and videos. As much as possible, the DLNA Guidelines refer to standards from established, open industry standards organizations such as the UPnP™ Implementers Corporation (UIC) and the Wi-Fi™Alliance. The DLNA Guidelines govern the design, compliance testing, and certification of new digital devices authorized to use the DLNA CERTIFIED™ logo and trademark. These Guidelines add additional restrictions and mandate the use of specific codecs to a number of standard protocols. Note: An implementation of this protocol will be included in Windows 7 Home Server.

Directory Services Markup Language 2.0: Directory Services Markup Language 2.0 defines methods for expressing directory queries and updates (and the results of these operations) using the SOAP protocol. It also provides a syntax for representing directory information as an XML document.

Discard Protocol standard: The Discard Protocol standard is a debugging and measurement tool. A discard service throws away any data that it receives.

Domain Name System (DNS) Protocol standard: The Domain Name System (DNS) is a client/server protocol standard used for name resolution, and for querying and updating name records. This includes responding to queries per the IETF standard DNS query protocol and accepting updates per the IETF standard DNS dynamic update protocol.

Domain Name System (DNS) Resource Record (RR) for Encoding Dynamic Host Configuration Protocol (DHCP) Information (DHCID RR): The Domain Name System's database is a collection of Resource Records. Each Resource Record specifies information, about a particular object. This protocol stores client identifiers in DNS to unambiguously associate domain names with DHCP clients, using a Dynamic Host Configuration ID (DHCID) RR. It defines mechanisms to update the FQDN records and minimizes chances of new clients registering themselves with DNS, using hostnames, that have been assigned to other servers.

Domain Name System Security Extensions (DNSSEC): Domain Name System Security Extensions (DNSSEC) are extensions to the Domain Name System (DNS) that provide data integrity and authentication to security aware resolvers or applications through the use of cryptographic digital signatures.

Dynamic Host Configuration Protocol (DHCP): The Dynamic Host Configuration Protocol (DHCP) standard provides a framework for passing configuration information to hosts on a TCP/IP network. DHCP is based on the Bootstrap Protocol (BOOTP), adding the capability of automatic allocation of reusable network addresses and additional configuration options. Some aspects of this protocol standard (including the DHCP relay agent) are not implemented in client SKUs.

Dynamic Host Configuration Protocol for IPv6 (DHCPv6): The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) standard enables DHCP servers to pass configuration parameters such as IPv6 network addresses to IPv6 nodes. It offers the capability of automatic allocation of reusable network addresses and additional configuration flexibility.

Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option: This protocol standard specifies a new Dynamic Host Configuration Protocol for IPv6

(DHCPv6) option that can be used to exchange information about a DHCPv6 client's Fully Qualified Domain Name (FQDN) and about responsibility for updating DNS resource records (RRs) related to the client's address assignments.

Echo Protocol: The Echo Protocol standard is a debugging and measurement tool. An echo service sends any data it receives back to the originating source. The data may be sent over either TCP or UDP.

Extensible Authentication Protocol (EAP) Message Digest Algorithm 5: The Extensible Authentication (EAP) standard is an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP. EAP Message Digest Algorithm 5 (EAP-MD5) is an EAP method supporting MD5-Challenge authentication. The MD5-Challenge method is analogous to the PPP CHAP protocol defined in RFC 1994, with MD5 as the specified algorithm.

Extensible Authentication Protocol Transport Level Security (EAP-TLS): The Extensible Authentication Protocol (EAP) standard is an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP. EAP Transport Layer Security (EAP-TLS) is an EAP method supporting mutual certificate-based authentication, based on TLS, specified in RFC 2246.

Extensible Authentication Protocol (EAP): The Extensible Authentication Protocol (EAP) standard is an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP. EAP provides its own support for duplicate elimination and retransmission, but is reliant on lower layer ordering guarantees. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this.

eXtensible Markup Language (XML): eXtensible Markup Language (XML) is a standard for formating streams of data. XML is stored within files and is used for some protocols as the descriptive language for the payload.

File Transfer Protocol (FTP): The File Transfer Protocol (FTP) standard is used to transfer files between two remote computers. FTP allows a remote computer to execute data transfer functions and file transfer functions on a remote host.

File Transfer Protocol (FTP) Extensions: These extensions to the File Transfer Protocol (FTP) enable support for FTP over Transport Layer Security (TLS), internationalization and the ability to specify a virtual host.

FTP extensions for IPv6 and NATs: The FTP extensions for IPv6 and NATs allow the FTP protocol to work over IPv6 and future network protocols. As it stands, without the extensions, FTP would not support IPv6 addresses.

General Event Notification Architecture (GENA): This specification provides for the ability to send and receive notifications using HTTP over TCP/IP and administratively scoped unreliable multicast UDP. Provisions are made for the use of intermediary arbiters, called subscription arbiters, which handle routing notifications to their intended destination.

Generic Security Services for Remote Procedure Call (RPCSEC_GSS): Generic Security Services for Remote Procedure Call (RPCSEC_GSS) is an Open Network Computing / Remote Procedure Call (ONC/RPC) security flavor that allows RPC protocols to access the Generic Security Services Application Programming Interface (GSS_API).

H.245 Protocol: H.245 specifies syntax and semantics of terminal information messages as well as procedures to use them for in-band negotiation at the start of or during communication. The messages cover receiving and transmitting capabilities as well as mode preference from the receiving end, logical channel signaling, and Control and Indication. Acknowledged signaling procedures are specified to ensure reliable audiovisual and data communication.

H.323 Protocol: The H.323 protocol standard describes protocols and packets between terminals and other entities that provide multimedia communications services over Packet Based Networks (PBN) which may not provide a guaranteed Quality of Service. H.323 entities may provide real-time audio, video and/or data communications. It is a protocol for establishing audio/video sessions and packet payloads sent and received during a session. Support for audio is mandatory, while data and video are optional, but if supported, the ability to use a specified common mode of operation is required, so that all terminals supporting that media type can interwork. The packet based network over which H.323 entities communicate may be a point-to-point connection, a single network segment, or an internetwork having multiple segments with complex topologies. H.323 entities may be used in point-to-point, multipoint, or broadcast (as described in Recommendation H.332) configurations.

High-Definition Multimedia Interface Audio Protocol: High-Definition Multimedia Interface (HDMI) is used to stream audio and video over a single cable from a source product (for example, PC, DVD player) to a connected device (for examle, TV, Audio/Video receiver). HDMI enables a source to query the connected device for its capabilities and send up to 8-channel, 192 kHz uncompressed audio in addition to compressed audio formats. Windows uses HDMI in order to retrieve the audio capabilities of the connected device and stream audio to that device.

Host Name Data Structure: RFC 952, as updated by RFC 1123, defines the syntax of a Host Name, which can be carried in various protocols such as DNS. RFC 2181 defines the behavior of DNS, which deals with generic DNS names (of which host names are just one type).

HTTP Authentication: Basic and Digest Access Authentication Protocol standard: The HTTP Authentication: Basic and Digest protocol standard verifies that both parties of a communication know a shared secret. This is commonly used to authenticate clients using browsers. This is the entry corresponding to the digest authentication implemented by digest.dll. Digest.dll no longer ships in the OS, as it was removed for Windows Vista. We have another for the Windows OS version wdigest.dll, which is the one replacing digest.dll. Basic authentication is implemented directly in WinInet and WinHTTP.

Human Interface Device Profile: The Bluetooth Human Interface Devices (HID) protocol standard defines a set of services that can be used between a host capable of supporting HID devices and a BT-HID device. Examples of this are mouse and keyboard.

Hypertext Markup Language (HTML) 4.01: This specification defines the HyperText Markup Language (HTML), the publishing language of the World Wide Web. This specification defines HTML 4.01, which is a subversion of HTML 4. In addition to the text, multimedia, and hyperlink features of the previous versions of HTML (HTML 3.2 [HTML32] and HTML 2.0 [RFC1866]), HTML 4 supports more multimedia options, scripting languages, style sheets, better printing facilities, and documents that are more accessible to users with disabilities. HTML 4 also takes great strides towards the internationalization of documents, with the goal of making the Web truly World Wide. This entry is intended to cover HTML 4.0 as well as HTML 4.01.

Hypertext Markup Language (HTML) 3.2: The HyperText Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. This specification defines HTML version 3.2. HTML 3.2 aims to capture recommended practice as of early '96 and as such to be used as a replacement for HTML 2.0 (RFC 1866).

Hypertext Transfer Protocol (HTTP): HTTP is an application-level, request/response protocol standard built on top of the TCP protocol. The protocol is standardized and maintained by the IETF and is currently at version 1.1.

ICMP Router Discovery Protocol: The ICMP Router Discovery Messages protocol standard specifies an extension of the Internet Control Message Protocol (ICMP) to enable hosts attached to multicast or broadcast networks to discover the IP addresses of their neighboring routers. The client role is implemented by the TCP/IP stack, and the server role is implemented by RRAS prior to Vista and by the TCP/IP stack in Vista.

IEC 61883: The IEC 61883 protocol standard uses the IEEE 1394 standard to specify a digital interface for electronic audio/video equipment.

IEEE 1394: IEEE 1394 multimedia connection enables simple, low-cost, high-bandwidth isochronous (real-time) data interfacing between computers, peripherals, and consumer electronics products such as camcorders, VCRs, printers, PCs, TVs, and digital cameras.

IEEE 802.11-2007 Protocol: This protocol standard is an implementation the Institute of Electrical and Electronic Engineers (IEEE) 802.11 wireless Local Area Network (LAN) specification. The protocol is used to provide a layer 2 connection between PCs, devices and other networks using unlicensed RF spectrum through a defined Media Access Control (MAC) and Physical layer (PHY). The protocol includes a client/server function as well as a peer to peer function. Note: Client acting as a server and Server acting as a server are only implemented in Windows 7 and Windows Server 2008 R2.

IEEE 802.1x-2004 (802.1x): This supplement to ISO/IEC 15802-3:1998 (IEEE Std 802.1D-1998) defines the changes necessary to the operation of a MAC Bridge in order to provide switch/bridge port-based network access control capability. For information on integrity checks and key encryption calculations for 802.1x, see the radius extensions RFC ftp://ftp.rfc-editor.org/in-notes/rfc3580.txt. Windows implements the client role only.

IEEE 802.2 (Logical Link Control): A set of procedures that are defined for the transfer of information and control between any pair of data link layer service access points on a Local Area Network (LAN). These procedures are independent of the type of medium access method used in the particular LAN. Windows includes third party drivers which implement the protocol for communication with their devices. These are in the form of drivers for Network Interface Cards. The IEEE 802.2 specification can be obtained from http://standards.ieee.org/getieee802/802.2.html.

IEEE 802.5: 802.5 (Token ring) is a protocol data link layer protocol standard that enables transfer of information and control between a pair of access points on a 802.5 Local Area Network (LAN). Down level Windows (pre Vista) includes third party drivers which implement the protocol for communication with their devices. These are in the form of drivers for Network Interface Cards. The IEEE 802.5 specification can be obtained from http://standards.ieee.org/getieee802/802.5.html.

Information Card Data Structure: Information Card is an XML document that contains data describing the relationship with a Security Token Service (STS) with respect to the token issuance of the STS. This data includes, among others, logical identifier for the issuer, token issuance endpoints of the STS, token types and claims supported by the STS, and the descriptor for the authenticating credential.

Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6): The Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 standard implements a Dynamic Host Configuration Protocol for IPv6 (DHCPv6) option for specifying an upper bound for how long a client should wait before refreshing information retrieved from DHCPv6. It is used with stateless DHCPv6 as there are no addresses or other entities with lifetimes that can tell the client when to contact the DHCPv6 server to refresh its configuration.

Integrated Services Digital Network (ISDN): The Integrated Services Digital Network (ISDN), as specified in ITU-T I.430 and ITU-T I.431, is a set of physical layer protocols for establishing and breaking circuit switched connections between the ISDN modem and the Internet Service Provider's telecommunication network. ISDN is designed to allow digital transmission of voice and data over ordinary telephone copper wires, resulting in better quality and higher speeds than that available with the PSTN system. Windows includes third party drivers which implement the ISDN set of protocols for communication with the telecommunications network. These are in the form of drivers for Connection oriented Network Interface Cards.

Interface - Parallel (IEEE 1284): A signaling method for asynchronous, fully interlocked, bidirectional parallel communications between hosts and printers or other peripherals is defined. A format for a peripheral identification string and a method of returning this string to the host outside of the bi-directional data stream is also specified. Specifications for Interface - Parallel (IEEE 1284) are maintained by the Institute of Electrical and Electronics Engineers, Inc., (IEEE). For information on Interface - Parallel (IEEE 1284) see http://standards.ieee.org/reading/ieee/std_public/description/busarch/1284-1994_desc.html.

Internet Control Message Protocol (ICMP): ICMP is used by Internet nodes and routers to signal error conditions, as well as for network diagnostics. ICMP is an integral part of the Internet Protocol Version 4 (IPv4).

Internet Control Message Protocol for IPv6 (ICMPv6): The Internet Protocol, version 6 (IPv6) standard is a new version of IP. ICMPv6 is the IPv6 equivalent of the Internet Control Message Protocol (ICMP) as defined for IPv4 [RFC-792]. The protocol is used to provide control notifications about network events. The mentioned RFCs also cover IPv6 neighbor and router discovery which are implemented on top of ICMPv6 and extensions to them (such as default router selection).

Internet Gopher Protocol: The Internet Gopher protocol standard is designed for distributed document search and retrieval via TCP port 70.

Internet Group Management Protocol MIB (IGMP MIB): This module exposes counters and data for tracking multicast sessions, using the Simple Network Management Protocol (SNMP). The IGMP MIB information is used by both the agent, that exposes the MIB information, and the MIB browser, which renders it. The IGMP MIB information ships on all client operating systems as part of the browser component, and the agent component shipped on Windows 2000 and Windows XP, but was removed from Windows Vista onwards.

Internet Group Management Protocol Version 1 (IGMPv1): The Internet Group Management Protocol (IGMP) standard is used by IP hosts to report their host group memberships to any immediately-neighboring multicast routers. It is also used by routers to query hosts for their membership in multicast groups. IGMP is an integral part of IP. It is required to be implemented by all Internet nodes. NOTE: RFC title is Host Extensions for IP Multicasting. IGMP is specified in Appendix A of the RFC. The RFC also deals with the data communication of IP multicasting in addition to the membership query and reporting capability. (The rest of the RFC is included as part of the IPv4 protocol entry.) Note that clients do listen to traffic from other clients for the purpose of suppressing their own messages.

Internet Group Management Protocol Version 2 (IGMPv2): IGMPv2 is used by IP hosts to report their multicast group memberships to multicast routers. IGMPv2 allows group membership termination to be quickly reported to the routing protocol, which is important for high-bandwidth multicast groups and/or subnets with highly volatile group membership. Note that clients do listen to traffic from other clients for the purpose of suppressing their own messages.

Internet Group Management Protocol Version 3 (IGMPv3): IGMP is the protocol standard used by IPv4 systems to report their IP multicast group memberships to neighboring multicast routers. Version 3 of IGMP adds support for "source filtering", that is, the ability for a system to report interest in receiving packets *only* from specific source addresses, or from *all but* specific source addresses, sent to a particular multicast address. That information may be used by multicast routing protocols to avoid delivering multicast packets from specific sources to networks where there are no interested receivers.

Internet Information Services (IIS) HTTP Protocol: The Hypertext Transfer Protocol (HTTP) standard is an application-level protocol standard for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

Internet Key Exchange (IKE): The Internet Key Exchange protocols that are used to negotiate IPsec settings and cryptographic keys in order to protect IP traffic from modification and information disclosure. (IPsec is a suite of protocols that protect network communication at the IP layer - Layer 3 in the OSI model. The main protocols for the application of security to IP packets are the AH and ESP protocols. AH and ESP in turn can use different encryption and hashing algorithms that constitute a transformation protocol.) RFC3947 was shipped first on Windows XP.

Internet Key Exchange Version 2 (IKEv2) Protocol: The Internet Key Exchange Version 2 (IKEv2) protocol standard is used to negotiate IP Security (IPsec) settings and cryptographic keys in order to protect Internet Protocol (IP) traffic from modification and information disclosure. The IKEv2 protocol includes the mobility and multihoming extension to Internet Key Exchange Version 2 (MOBIKE). MOBIKE allows the IP addresses associated with Internet Key Exchange Version 2 (IKEv2) Protocol and tunnel mode IP Security Security Associations (SAs) to change.

Internet Message Access Protocol - Version 4: IMAP4 is a protocol standard for retrieving e-mail messages from a server. The client side ships as part of Outlook Express and the server side is part of Exchange Server.

Internet Printing Protocol: The Internet Printing Protocol (IPP) is a standard, application-level protocol standard that can be used for distributed printing using Internet tools and technologies. IPP uses a simplified model consisting of abstract objects, their attributes, and their operation.

Internet Protocol over Asynchronous Transfer Mode (IPoATM) Protocol: The IP over ATM protocol standard defines procedures to resolve IP addresses to ATM addresses, and to carry IP datagrams within ATM Adaptation Layer 5 (AAL5) data units within ATM Virtual Circuits (VCs) and thus provide IP connectivity over an ATM network. Third party vendors provide device drivers that transmit and receive AAL5 data units through ATM Network Interface Cards. An ATM ARP server is the only server role we implement. We do not implement an ATMUNI switch role.

Internet Protocol Security (IPsec & IKE): Cryptographic Algorithms: A set of IETF standards track RFCs define the IPsec protocol. IPsec protects network communication at the IP layer - Layer 3 in the OSI model. The main protocols for the application of security to IP packets are the Authentication Header (AH) and Encapsulated Security Payload (ESP) protocols. AH and ESP in turn can use different encryption and hashing algorithms that constitute a payload transform. The microsoft implementation of IPsec supports Advanced Encryption Standard (AES) Gaulois counter mode Message Authentication Code (GMAC) and AES Cipher Block Chaining Counter Mode (CCM) for use with AH and ESP.

Internet Protocol Security (IPsec) Protocol: The (IETF standard) IPsec protocols that are used to protect IP traffic from modification and information disclosure. IPsec is a suite of protocols that protect network communication at the IP layer - Layer 3 in the OSI model. The main protocols for the application of security to IP packets are the Authentication Header (AH) and Encapsulation Security Protocol (ESP) protocols. AH and ESP in turn can use different encryption and hashing algorithms that constitute a transformation protocol.

Internet Protocol Version 4 (IPv4): The Internet Protocol standard is designed for use in interconnected systems of packet-switched computer communication networks. The Internet Protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The Internet Protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks. The IP extensions for IP packet multicasting are described in RFC 1112.

Internet Protocol Version 6 (IPv6): The IPv6 protocol standard is used by nodes to communicate with other nodes across a network, independent of the types of physical links present in the network. IPv6 is the successor to IPv4 (RFC 791). The Internet Protocol (IP) is a data-oriented protocol used for communicating data across a packet-switched internetwork. IP is a network layer protocol in the internet protocol suite and is encapsulated in a data link layer protocol (e.g., Ethernet). As a lower layer protocol, IP provides the service of communicable unique global addressing amongst computers. This implies that the data link layer need not provide this service. This form also equally applies to ICMPv6 and IPv6 stateless address auto configuration (RFC2463, RFC2462). The Microsoft IPv6 implementation also supports the host part of the router alert option RFC (RFC2711).

Internet SCSI (ISCSI) Protocol: The iSCSI protocol standard transports the SCSI storage protocol (T10 SPC-2 or greater) over TCP/IP. The protocol enables access from a computer to SCSI devices, such as disks, tapes, changers, and CD ROMs. It defines the rules and processes for device discovery, authentication, and data transfer. The protocol also provides support for a session (nexus) to implement fault tolerant access to disks across the network.

Internet Storage Name Service (iSNS): The Internet Storage Name Service (iSNS) Protocol standard is used for interaction between iSNS servers and iSNS clients, and facilitates automated discovery, management, and configuration of iSCSI and Fibre Channel devices (using iFCP gateways) on a TCP/IP network. iSNS provides intelligent storage discovery and management services comparable to those found in Fibre Channel networks, allowing a commodity IP network to function in a capacity similar to that of a storage area network. iSNS facilitates a seamless integration of IP and Fibre Channel networks due to its ability to emulate Fibre Channel fabric services and to manage both iSCSI and Fibre Channel devices. iSNS thereby provides value in any storage network comprised of iSCSI devices, Fibre Channel devices (using iFCP gateways), or any combination thereof.

Internet X.509 Public Key Infrastructure Time-Stamp Protocol: This protocol standard is a transport-agnostic Cryptographic Message Syntax (CMS) based request-reply protocol for requesting and receiving a cryptographically signed time stamp token in response to the provision of a cryptographic hash. This time stamp is used as evidence that the hashed object existed prior to the time given in the time stamp.

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP): The Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) standard provides IPv6 connectivity across an IPv4 intranet. ISATAP views the IPv4 network as a link layer for IPv6 and views other nodes on the network as potential IPv6 hosts/routers. ISATAP supports an automatic tunneling abstraction similar to the Non-Broadcast Multiple Access (NBMA) model.

IP Forwarding Table MIB: This MIB module defines settings for managing routes in an IP network.

IP Version 6 over PPP: This protocol standard defines a method for transmission of IP Version 6 packets over PPP links. It also specifies the method of forming IPv6 link-local addresses on PPP links.

IP-in-IP Tunneling: This is a connectionless protocol standard used for point-to-point tunnels between routers, to construct overlay networks. Support for this was added to RRAS in Windows 2000 Server, and was then removed from subsequent releases.

IPv4 Multicast Routing MIB: This MIB module provides the ability to read counters related to the IPv4 Multicast Routing protocol standard using SNMP.

IPv4 over High Performance Serial Bus (IEEE 1394) Protocol: This protocol standard specifies how to use a High Performance Serial Bus (IEEE 1394), for the transport of Internet Protocol Version 4 (IPv4) datagrams. This includes not only packet formats and encapsulation methods for datagrams, but also an address resolution protocol (1394 ARP) and a multicast channel allocation protocol (MCAP) that permits management of Serial Bus resources when used by IP multicast groups. This protocol is no longer supported as of Windows Vista.

IPv6 MIB: This MIB module specifies configuration settings for the IPv6 protocol.

IPv6 over IPv4 (6over4) Protocol: 6over4, also known as IPv4 multicast tunneling, is a host-to-host, host-to-router, and router-to-host automatic tunneling technology that is used to provide unicast and multicast IPv6 connectivity between IPv6 nodes across an IPv4 intranet.

IPv6-in-IPv4 Configured Tunnels: IPv6-in-IPv4 Configured Tunnels, as defined in a portion of RFC 2893, are point-to-point tunnels made by encapsulating IPv6 packets within IPv4 headers to carry them over IPv4 routing infrastructures. Windows supports the ability to manually configure such IPv6 tunnels via netsh.

IrDA Infrared Transfer Picture (IrTranP) Protocol: IrTranP is an image transfer protocol standard over the Infrared link for digital cameras. IrTranP includes a binary file transfer protocol and a simple command execute protocol to transfer a Uni Picture Format (UPF). IrTranP is a higher layer protocol sitting on top of Ir-Tiny-TP, Ir-LMP, Ir-LAP, which collectively constitute the Infrared communications stack.

IrDA Tiny Transport Protocol (Tiny TP): The Tiny TP protocol standard provides independently flow controlled transport connections over the IrLMP (Infrared Link Management Protocol). In other words, IrTinyTP abstracts the lower layers of the Infrared stack (IrLMP and IrLAP - Link Access Protocol) from upper layer protocols by means of providing abstracted methods to utilize the Infrared stack. The abstract methods are 'transport connections' which represent one flow of data from (or to) an application or application layer protocol. The IrTinyTP protocol is part of the 'core' Infrared protocols which are implemented in Windows as part of the native support in Windows for Infrared.

JavaScript Object Notation Protocol: JSON is a data interchange format based on the Ecmascript programming language. Our usage in ASP.NET is similar to using XML between a Web client and server for transmitting data. More about JSON can be found at http://www.json.org.

Kerberos Network Authentication Service (V5): The Kerberos authentication protocol standard allows a user to log in to a network service, the KDC (key distribution center / Kerberos domain controller). Once a user is authenticated to the KDC, a session key is generated that is used by that user's machine for subsequent authentications to any computer or service in the same Kerberos domain or any other Kerberos domain that cooperates with ("shares trust with") the user's domain. When used for network authentication, the protocol is 3-way - allowing client-to-client authentication, intermediated by the KDC server. This implies that there are two different kinds of "server" role. In the entries in this form, we assume only the KDC implements the "key server role" while any computer can implement the "app server role".

Kermit File Transfer Protocol: Kermit is a file transfer protocol standard developed by Columbia University beginning in 1981.The Kermit site may be found at: http://www.columbia.edu/kermit/index.html. The protocol is documented in the book: Kermit: A File Transfer Protocol by Frank Da Cruz, Digital Press 1987, ISBN 0932376886. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_kermit_file_transfer.asp.

LAN Emulation over ATM (LANE) Protocol: This protocol standard supports the emulation of an IEEE 802.3 (Ethernet) or IEEE 802.5 (Token Ring) Local Area Network over an underlying Asynchronous Transfer Mode (ATM) network, for the purposes of operating protocols such as IP and/or IPX over such a network. The protocol includes services for configuring end-nodes (hosts), mapping between Ethernet/Token Ring addresses and ATM addresses and transport of unicast, broadcast and multicast Ethernet/Token Ring frames. This provides Ethernet to ATM address resolution and Ethernet frame encapsulation and transport over ATM. The server role is an ATM switch. Communication between Windows client and server products is always goes through an ATM switch.

Lightweight Directory Access Protocol (LDAP): This is a directory access protocol standard that provides for the ability to create, read, update, and delete entries in a directory service such as Microsoft Active Directory. Each entry is an object identified by a unique name and containing one or more attributes. Each of those attributes can have one or more values. Although the title does not specify, this is LDAP V2.

Lightweight Directory Access Protocol (LDAP) Version 3: This is a directory access protocol standard that provides the ability to create, read, update, and delete entries in a directory service such as Microsoft Active Directory. This protocol standard builds upon LDAP V2 as its foundation, but adds greater extensibility via the LDAP extended control and LDAP extended operation mechanisms, as well as refining and expanding upon other concepts initially introduced in LDAP v2. This is the preferred version of LDAP to use in accessing Active Directory. The following Active Directory MMC snap-ins make use of the LDAP V3 protocol standard: dsa.msc, dssite.msc, domain.msc, schmgmt.msc, adsiedit.msc, gpedit.msc, certmgr.msc, rsop.msc. Tri-state filter evaluation was not implemented in Win2k RTM; therefore Win2K only implements the client role.

Lightweight Online Certificate Status Protocol (OCSP): This protocol standard, specified in RFC 5019 as "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High Volume Environments", provides revocation information for x.509 certificates. It allows applications to query the current status of a certificate without downloading a certificate revocation list (CRL), which can be large in size. RFC 5019 is a profile of the Online Certificate Status Protocol (OCSP) specified in RFC 2560.

Line Printer Daemon Protocol: The Line Printer Daemon (LPD) protocol standard communicates between line printer daemons (clients and servers). The Berkeley versions of the Unix operating system provide line printer spooling with a collection of programs: lpr (assign to queue), lpq (display the print queue), lprm (remove from print queue), and lpc (control the print queue). These programs interact with an autonomous process called the line printer daemon.

Management Information Base for Network Management of TCP/IP-based internets (MIB-II): This MIB module defines settings for the Internet Protocol Version 4 (IPv4).

Media Transfer Protocol (MTP): Media Transfer Protocol, or MTP, is a protocol standard that is designed for content exchange with and command and control of transient storage devices. The primary purpose of this protocol standard is to facilitate communication between media devices that have transient connectivity and significant storage capacity. This includes the exchange of binary objects and the enumeration of the contents of that connected device. The secondary purpose of this protocol standard is to enable command and control of the connected device. This includes the remote invocation of device functionality, monitoring of device-initiated events, and the reading and setting of device properties. A server in MTP serves the role of initiating the connection and sending the commands. A server in MTP is the MTP Initiator. A client is an MTP responder.

Microsoft Point-to-Point Compression (MPPC) Protocol: The Point-to-Point Protocol (PPP) provides a standard method for transporting multi-protocol datagrams over point-to-point links. The PPP Compression Control Protocol provides a method to negotiate and utilize compression protocols over PPP encapsulated links. The Microsoft Point-to-Point Compression (MPPC) protocol describes a method compressing PPP encapsulated packets.

Microsoft Point-To-Point Encryption (MPPE) Protocol: The Point-to-Point Protocol (PPP) standard provides a standard method for transporting multi-protocol datagrams over point-to-point links. The PPP Compression Control Protocol provides a method to negotiate and utilize compression protocols over PPP encapsulated links. The Microsoft Point-To-Point Encryption (MPPE) Protocol describes a means for enhancing the confidentiality of PPP-encapsulated packets, through encryption.

Microsoft PPP CHAP Extensions: This protocol standard describes Microsoft's PPP CHAP dialect (MS-CHAP), which extends the user authentication functionality provided on Windows networks to remote workstations. MS-CHAP is closely derived from PPP Challenge Handshake Authentication Protocol described in RFC 1994.

Microsoft PPP CHAP Extensions, Version 2: This protocol standard defines version 2 of Microsoft's PPP CHAP dialect (MS-CHAP-V2). MS-CHAP-V2 is similar to, but not compatible with, MS-CHAP version 1 (described in RFC). In particular, some protocol fields have been deleted or reused with different semantics. In addition, MS-CHAP-V2 provides for mutual authentication between the client and the server.

Minitel Terminal Emulation: Minitel was developed to work with dumb terminals attached to servers on an X.25 network via the V.23 modem protocol described in: International Telecommunication Union, "600/1200-baud modem standardized for use in the general switched telephone network", ITU-T Recommendation V.23, November 1988. Further information on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_minitel_emulation.asp

Moving Picture Experts Group (MPEG) Stream Encoding Protocol: The Moving Picture Experts Group (MPEG) Stream Encoding Protocol standard is used to encode and to stream broadcast audio/video content and related data from a device or service. Windows contains multimedia components for supporting the client side of the mpeg-2 transport stream (TS) and Packetized Elementary Stream (PES) stream protocols. This protocol is used in the Windows client and by Media Center in the Windows 7 Entertainment Server SKUs.

Multicast Address Dynamic Client Allocation Protocol (MADCAP): The Multicast Address Dynamic Client Allocation Protocol (MADCAP) standard allows hosts to request multicast address allocation services from multicast address allocation servers.

Multicast Listener Discovery Version 1 (MLDv1): This protocol standard is used by an IPv6 router to discover the presence of multicast listeners (that is, nodes wishing to receive multicast packets) on its directly attached links, and to discover specifically which multicast addresses are of interest to those neighboring nodes. MLD is derived from version 2 of IPv4's Internet Group Management Protocol, IGMPv2. Note that clients do listen to traffic from other clients for the purpose of suppressing their own messages.

Multicast Listener Discovery Version 2 (MLDv2): This protocol standard specifies Version 2 of the Multicast Listener Discovery Protocol (MLDv2). MLD is used by an IPv6 router to discover the presence of multicast listeners on directly attached links, and to discover which multicast addresses are of interest to those neighboring nodes. MLDv2 is designed to be interoperable with MLDv1. MLDv2 adds the ability for a node to report interest in listening to packets with a particular multicast address only from specific source addresses or from all sources except for specific source addresses.

Multipurpose Internet Mail Extensions (MIME): MIME is short for Multipurpose Internet Mail Extensions, a specification for formatting non-ASCII messages so that they can be sent over the Internet. This implementation of the parser for this file format ships as part of Outlook Express.

NetBIOS over TCP (NetBT) Protocol: The NETBT protocol standard supports NetBIOS services in a TCP/IP environment.

Network File System (NFS) Protocol: Network File System (NFS) version 2 and 3 is a protocol standard originally developed by Sun Microsystems in 1984 and defined in RFCs 1094 (version 2), 1813 (version 3) as a distributed file system which allows a computer to access files over a network as easily as if they were on its local disks. It is built upon two other standards - XDR (RFC 1014) and Sun RPC (RFC 1057). Embedded within RFC 1813, Appendix II, is the Lock Manager Protocol, which often is thought of as a separate protocol, and is also supported. This protocol standard originally shipped in Microsoft Windows Services for Unix, and was later integrated in the client with Vista and into the server with Windows Server 2003 and later server SKUs (NFS client on client SKUs, NFS client and server on server SKUs).

Network News Transfer Protocol: Exchange of messages in a public forum across the internet - network news transport protocol. The NNTP Server does not ship in Windows Vista client or Windows Longhorn server.

Network Time Protocol (Version 3): Specification, Implementation and Analysis: The Network Time Protocol (NTP) provides the mechanisms to synchronize time and coordinate time distribution in a large, diverse internet operating at rates from mundane to lightwave. It uses a returnable-time design in which a distributed subnet of time servers operating in a self-organizing, hierarchical-master-slave configuration synchronizes local clocks within the subnet and to national time standards via wire or radio. The servers can also redistribute reference time via local routing algorithms and time daemons.

ONC Remote Procedure Call (RPC) Protocol: ONC RPC and XDR specify message communication and data transfer between different computer architectures using a remote procedure call style of data transfer. They are the required basis for the Network File System protocol. This protocol originally shipped in Services-for-Unix, and was later integrated in the client with Vista and into the server with Windows Server 2003 and later server SKUs (ONC RPC client on client SKUs, ONC RPC client and server on server SKUs).

Online Certificate Status Protocol (OCSP): This protocol standard, specified in RFC 2560 as "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", provides revocation information for X.509 certificates. It allows applications to query the current status of a certificate without downloading a certificate revocation list (CRL), which can be large in size.

Open Font Format - Font File Format: This OpenType Font File Format details the format of OpenType fonts, the TrueType and CFF outline formats, and the TrueType hinting language. The specification has been developed jointly by Adobe and Microsoft. Because OpenType is an extension of TrueType, portions are also developed jointly by Adobe, Microsoft, and Apple. OpenType version 1.4 has been contributed to ISO and became the foundation for the development of ISO/IEC 14496-22 "Open Font Format" standard. The standard was published in 2007, and is now freely available for download from ITTF website. OpenType version 1.5 was recently published and reflects updates to OpenType implemented within Windows Vista, as well as fonts issued by Adobe. OpenType 1.6 is in development and will contain future (post Windows 7) enhancements. Updates to the specification have been submitted for inclusion in ISO/IEC 14496-22, the "Open Font Format". These changes are reflected in the Committee Draft of the second edition of the ISO/IEC 14496-22.

OpenSearch Version 1.1 File Formats: The OpenSearch Version 1.1 file formats are a collection of simple formats for the sharing of search results over Hypertext Transfer Protocol. The OpenSearch description document format defines the URL format used in querying a search engine. OpenSearch also defines additional elements to include in RSS or Atom result formats to help clients consume a search server's results.

OpenSearch Version 1.1 Protocol: The OpenSearch protocol standard defines the way in which a search query URL should be formed by the search client for sending to a search server and how the client should interpret specific parameters returned in the resulting response sent back from the server.

OSPF Version 2: OSPF version 2 is a link-state routing protocol. It is designed to be run internal to a single Autonomous System. Each OSPF router maintains an identical database describing the Autonomous System's topology. From this database, a routing table is calculated by constructing a shortest-path tree. OSPF recalculates routes quickly in the face of topological changes, utilizing a minimum of routing protocol traffic. OSPF provides support for equal-cost multipath. An area routing capability is provided, enabling an additional level of routing protection and a reduction in routing protocol traffic. In addition, all OSPF routing protocol exchanges are authenticated.

Part 12: ISO Base Media File Format (MPEG-4): ISO/IEC 14496-12:2005 specifies the structure and uses of the ISO base media file format. The identical text is published as ISO/IEC 15444-12:2005. This file format is used to contain time-based media such as video and audio. The storage of particular coding schemes is defined in specifications that derive from and reference ISO/IEC 14496-12:2005 and ISO/IEC 15444-12:2005, such as the MPEG-4 file format specified in ISO/IEC 14496-14, or the Motion JPEG file format specified in ISO/IEC 15444-3. This file format is designed to contain timed media information for a presentation in a flexible, extensible format that facilitates interchange, management, editing and presentation of the media. This presentation may be "local" to the system containing the presentation, or may be via a network or other stream delivery mechanism. The file format is designed to be independent of any particular network protocol while enabling efficient support for them in general. The file structure is object-oriented; a file can be decomposed into constituent objects very simply, and the structure of the objects inferred directly from their type. This technically identical text is published as ISO/IEC 14496-12:2005 for MPEG-4, and as ISO/IEC 15444-12:2005 for JPEG 2000. This version adds various new tools, including those for content protection, better support of metadata, and better support for advanced coding. This currently ships Out of Band with Windows Server 2008 supplemental EULA. This is available on Windows Server 2008 and Windows Server 2008 R2. This File Format can be used with a client that is shipped in Microsoft Expression Encoder. The client is shipped along with the source as a template.

Password Authentication Protocol (PAP): The Point-to-Point Protocol (PPP) standard provides a standard method of encapsulating Network Layer protocol information over point-to-point links. PPP also defines an extensible Link Control Protocol, which allows negotiation of an Authentication Protocol for authenticating its peer before allowing Network Layer protocols to transmit over the link. The Password Authentication Protocol is one of two protocols for authentication defined in RFC 1334. It passes the user name and password in plaintext to the server.

Picture Transfer Protocol (PTP): The primary purpose of Picture Transfer Protocol (PTP) is to provide a common protocol for any device, including digital photography devices, to exchange images with a DSPD, either by retrieving images from a digital camera or by sending images to or from a digital camera. Secondary purposes include a mechanism for devices to control digital cameras (e.g. a PC can request that a digital camera change its shutter duration setting and capture a new picture) and the ability to transfer auxiliary information such as non-image data files and associated information, such as a digital print order file (DPOF). A server is defined as a PTP Initiator by the PTP Specification. A client is defined as a PTP Responder by the PTP Specification.

Point to Point Protocol (PPP): The Point-to-Point Protocol (PPP) standard provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP is comprised of three main components:

- A method for encapsulating multi-protocol datagrams.
- A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.
- A family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.

Point-to-Point Protocol over ATM Adaptation Layer 5 (PPPoA): The Point-to-Point Protocol (PPP) standard provides a standard method for transporting multi-protocol datagrams over point-to-point links. This protocol describes the use of ATM Adaptation Layer 5 (AAL5) for framing Point-to-Point Protocol (PPP) encapsulated packets. PPPoA is used by a Windows client to connect to the Internet over an Asynchronous Digital Subscriber Line (ADSL) link.

Point-to-Point Protocol over Ethernet (PPPoE): The Point-to-Point Protocol (PPP) standard provides a standard method for transporting multi-protocol datagrams over point-to-point links. This protocol describes how to build PPP sessions and encapsulate PPP packets over Ethernet. PPPoE is used by a Windows client accessing the Internet over an Asynchronous Digital Subscriber Line (ADSL) link.

Portmap and RPCBind Protocol: Portmap and RPCBind are built on top of ONC RPC and enable discovery of ONC RPC – based services. Using the Portmap protocol standard, client computers discover the services, such as NFS, available on a server computer and their corresponding networking transports and IP ports. This protocol standard originally shipped in Services-for-Unix, and was later integrated in the client with Vista and into the server with Windows Server 2003 and later server SKUs (The client ships on client SKUs, the client and server on server SKUs).

Post Office Protocol - Version 3: The Post Office Protocol, version 3, (POP3) standard permits a workstation to dynamically access a mail drop on a server host. The POP3 protocol is used to allow a workstation to retrieve mail that the server is holding for it. The server side component will not ship with Windows 7 (it only shipped on Windows Server 2003 and Windows Server 2003 R2). Only the client side code will be present.

PPP AppleTalk Control Protocol (ATCP): The Point-to-Point Protocol (PPP) standard provides a standard method of encapsulating Network Layer protocol information over point-to-point links. PPP also defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols. This extension defines the NCP for establishing and configuring the AppleTalk Protocol over PPP.

PPP Bandwidth Allocation Protocol (BAP) / PPP Bandwidth Allocation Control Protocol (BACP): BAP provides a method to manage the dynamic bandwidth allocation of implementations supporting the PPP multilink protocol. BAP can be used to manage the number of links in a multilink bundle. BAP defines datagrams to co-ordinate adding and removing individual links in a multilink bundle, as well as specifying which peer is responsible for which decisions regarding managing bandwidth during a multilink connection.

PPP Challenge Handshake Authentication Protocol (CHAP): Challenge Handshake Authentication Protocol (CHAP) standard defines a method for authentication using PPP, which uses a random challenge, with a cryptographically hashed Response which depends on the Challenge and a secret key.

PPP Compression Control Protocol (CCP): The PPP Compression Control Protocol (CCP) standard configures, enables, and disables data compression algorithms on both ends of the point-to-point link.

PPP Internet Protocol Control Protocol (IPCP): The Point-to-Point Protocol (PPP) standard provides a standard method of encapsulating Network Layer protocol information over point-to-point links. PPP also defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols. This protocol standard defines the NCP for establishing and configuring the Internet Protocol over PPP, and a method to negotiate and use Van Jacobson TCP/IP header compression with PPP. In addition, IPCP Extensions extend the NCP for establishing and configuring the Internet Protocol over PPP, defining the negotiation of primary and secondary Domain Name System (DNS) and NetBIOS Name Server (NBNS) addresses.

PPP Internetwork Packet Exchange Control Protocol (IPXCP): The Point-to-Point Protocol (PPP) standard provides a method for transmitting multi-protocol datagrams over point-to-point links. PPP defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols for establishing and configuring different network-layer protocols. The IPX protocol was originally used in Novell's NetWare products, and was supported briefly by other vendors. This protocol defines a Network Control Protocol for establishing and configuring the IPX protocol over PPP.

PPP Multilink Protocol (MP): This protocol standard defines a method for splitting, recombining and sequencing datagrams across multiple logical data links. This work was originally motivated by the desire to exploit multiple bearer channels in ISDN, but is equally applicable to any situation in which multiple PPP links connect two systems, including asynchronous links. This is accomplished by means of new PPP options and this protocol.

PPP NetBIOS Frames Control Protocol (NBFCP): The Point-to-Point Protocol (PPP) standard provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols for establishing and configuring different network-layer protocols. The NBF protocol was originally called the NetBEUI protocol. NBFCP defines a Network Control Protocol for establishing and configuring the NBF protocol over PPP.

Pragmatic General Multicast (PGM) Protocol: The PGM Protocol standard is a reliable multicast transport protocol for applications that require ordered or unordered, duplicate-free, multicast data delivery from multiple sources to multiple receivers.

Printer MIB: This MIB module is used for management of printed document production and management of printer settings. Some OIDs defined by the Printer MIB are used to query printers for status and configuration.

Protocol Modifications for the DNS Security Extensions: The Domain Name System Security Extensions (DNSSEC) is a suite of extensions to the Domain Name System (DNS) protocol designed to provide origin authority of DNS data, data integrity, and authenticated denial of existence.

Public Key Cryptography Based User-to-User Authentication (PKU2U) Protocol: The Public Key Cryptography Based User to User Authentication Protocol (PKU2U) standard provides security services in peer to peer networking environments without requiring a Kerberos Key Distribution Center (KDC). A binding of this protocol for the Generic Security Service Application Program Interface (GSS-API) is provided.

Public Key Cryptography for Initial Authentication in Kerberos (PKINIT): The Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) protocol extensions enable the use of public key cryptography in the initial authentication exchange of the Kerberos protocol (Authentication Service (AS) exchange) [MS-KERB]. In the initial authentication exchange, Kerberos uses passwords shared between the client and the Key Distribution Center(KDC) to derive a key (AS-REP key) that is used to encrypt the Kerberos Ticket-Granting Ticket (TGT) and Ticket Granting Service (TGS) requests [RFC4120]. Using this scheme, the Kerberos protocol encryption strength is tied to the strength of the passwords used, and affects the security of subsequent protocol requests.

Quote of the Day Protocol: The Quote Of The Day (QOTD) service is an Internet protocol standard defined in RFC 865. It is intended for testing and measurement purposes. A host connects to a server that supports the QOTD protocol, on either TCP or UDP port 17. The server then returns a short arbitrary message. This was traditionally a random selection from a list of notable quotes.

Really Simple Syndication (RSS) 2.0: Really Simple Syndication (RSS) is an XML based file format that describes lists of related information known as feeds. An RSS document contains a <channel> element, with the information about the channel (metadata) and its content.

Real-Time Streaming Protocol (RTSP): The Real Time Streaming Protocol, or RTSP, is an application-level protocol standard for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP (RFC 1889). Note: an implementation of this protocol is included in Windows 7 Home Server.

Remote Access Dial In User Service (RADIUS) Accounting Server MIB (ACCSERV.MIB): The Remote Access Dial In User Service (RADIUS) Protocol standard defined in RFC 2865 provides Authentication, Authorization, and Accounting (AAA) of endpoints in scenarios such as wireless networking, dial-up networking and virtual private networking (VPN). These settings instrument RADIUS accounting server functions, enabling IP-based management stations to manage RADIUS accounting servers.

Remote Access Dial In User Service (RADIUS) Authentication Server MIB (AUTHSERV.MIB): The Remote Access Dial In User Service (RADIUS) Protocol standard defined in RFC 2865 provides Authentication, Authorization, and Accounting (AAA) of endpoints in scenarios such as wireless networking, dial-up networking and virtual private networking (VPN). These settings instrument RADIUS authentication server functions, enabling IP-based management stations to manage RADIUS authentication servers.

Remote Access Dial In User Service (RADIUS) IPv6: The Remote Access Dial In User Service (RADIUS) Protocol standard defined in RFC 2865 provides Authentication, Authorization, and Accounting (AAA) of endpoints in scenarios such as wireless networking, dial-up networking and virtual private networking (VPN). This protocol supports the operation of RADIUS over IPv6 and defines the RADIUS attributes used to support IPv6 network access.

Remote Authentication Dial-In User Service: Remote Authentication Dial-In User Service (RADIUS) protocol is commonly used to provide centralized authentication, authorization, and accounting for dial-up, virtual private network, and, more recently, wireless network access.

Resource ReSerVation Protocol (RSVP): The resource reservation setup (RSVP) protocol standard is designed for an integrated services (IntServ) network. RSVP provides receiver-initiated setup of resource reservations for multicast or unicast data flows. RSVP requires signaling cooperation from network elements (e.g., layer-3 routers). There are three roles in RSVP: The requestor of the reservation; the responding end point; intermediate routers. We implement the behaviors of the two end points. We do not implement the router behavior.

RIP Version 2 MIB: This MIB can be used to read counters related to the Routing Information Protocol version 2 (RIPv2) using SNMP on the RRAS server.

Rlogin Protocol: The rlogin facility provides a remote-echoed, locally flow-controlled virtual terminal with proper flushing of output. It is widely used between Unix hosts because it provides transport of more of the Unix terminal environment semantics than does the Telnet protocol, and because on many Unix hosts it can be configured not to require user entry of passwords when connections originate from trusted hosts. The protocol standard is requires the use of TCP and uses port 513.

Routing Information Protocol (RIP 1.0, 2.0): Routing Information Protocol standard is intended to do the following things:

- Define a protocol and algorithms that are used for routing.
- Specify improvements in the algorithms which will improve stability of the routes in large networks.
- Suggest some optional features to allow greater configurability and control.

RIP Version 2 specifies an extension of the Routing Information Protocol (RIP), as defined in, to expand the amount of useful information carried in RIP messages and to add a measure of security.

RS-232 Serial Protocol: The original RS-232-C standard defined the electrical characteristics of a serial communications link, including signal voltages, signal timings, and header pin mapping. Characters sent across the link are bracketed during transport by start and stop bits that allow the receiver to stay in synchrony with the sender. In current usage, the term "RS-232" is often used to mean "ASCII (or ANSI) serial protocol", and no longer indicates specific voltage levels. For a serial link, care should be taken that the signal voltages at the two ends of the link are compatible. For example, a PC COM port operating at a nominal 12 volts is incompatible with the serial connector of a microprocessor platform operating at 3 volts. For ANSI terminal emulation see the "ANSI Terminal Emulation" protocol entry. Further information on the MSDN website at: http://msdn.microsoft.com/library/en-us/randz/protocol/hyperterminal_rs-232_communications.asp.

SCSI Medium Changer Commands: This standard defines the SCSI commands and model for independent media changer devices and attached media changer functions integrated into other SCSI devices.

SCSI Multimedia Command Set: The SCSI Multimedia Command Set protocol standard allows communication with optical devices (for example, CD and DVD).

SCSI Primary Commands: The SCSI (Small Computer System Interface) Primary Commands defines the basic set of mandatory commands used with SCSI peripherals, such as hard drives, tape drives, scanners. These commands are used to drive device (storage, printer, scanner, optical) operation. Examples of the commands are Read, Write, Inquiry. Windows Server can virtualize storage devices and hence can act as a server role of the protocol.

SCSI Stream Commands: This standard specifies functional requirements for SCSI-3 Stream Commands (SSC). SSC permits SCSI streaming devices such as tape and printer devices to attach to computers and provides the definitions for their use.

Secure Digital Card Protocol: The Secure Digital (SD) Card protocol standard is used to support small form factor removable storage devices, like SD flash cards. The SD Card protocol enables communication between PC and locally attached storage cards over SD hardware bus within the PC. It enables discovery and mounting of SD card devices as removable file system volumes.

Security Assertion Markup Language (SAML) V1.1: The Security Assertion Markup Language (SAML) defines the syntax and processing semantics of assertions made about a subject by an issuer of assertions. SAML defines an XML data structure that may be used to describe attributes about a subject such as a user. The data structure is usually signed by its issuer and serves as a security token.

Security Assertion Markup Language (SAML) v2.0: The Security Assertion Markup Language (SAML) V2.0 defines the protocols for exchanging assertions made about a subject by an issuer of assertions, as well as the syntax and processing semantics of those assertions. SAML V2.0 defines an HTTP based block protocol for send assertions between servers using a Web browser intermediary. SAML V2.0 defines an XML data structure for assertions that may be used to describe attributes about a subject such as a user. The data structure is usually signed by its issuer and serves as a security token. This protocol will ship as an out of band (OOB) Windows Targeted Release (WTR) for Windows Server 2008 R2, Windows Server 2008, and Windows Server 2003. This protocol is a server SKU to server SKU exchange using any arbitrary Web browser as an intermediary. Client SKUs are not checked because, even though this protocol can be relayed through a client Web browser, there is no code for this protocol in the Web browser.

Serial Line Internet Protocol (SLIP): Serial Line IP is a protocol standard used for point-to-point serial connections running TCP/IP. It defines a standard encapsulation for IP packets over serial lines. This protocol standard is no longer in use and has been deprecated in Windows Vista and Longhorn.

Session Description Protocol (SDP): Session directories assist the advertisement of conference sessions and communicate the relevant conference setup information to prospective participants. SDP is designed to convey such information to recipients. SDP is purely a format for session description - it does not define or require a transport protocol.

Session Initiation Protocol (SIP): Session Initiation Protocol (SIP) standard is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. It is defined in RFC 2543 and later RFC 3261. The Real-Time Communications (RTC) API in Windows XP implemented RFC 2543. Windows Messenger uses the RTC API to use SIP to set up audio and video chat sessions with each other. A client is an MTP responder.

Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO): The Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) is used to select among possible authentication protocols. SPNEGO provides a framework for two parties engaged in authentication to select from a set of possible authentication mechanisms, in a fashion that preserves the opaque nature of the Security Protocols to the Application Protocol that uses SPNEGO. SPNEGO was first defined as RFC 2478. This RFC has since been supplanted by RFC 4178.

Simple Mail Transfer Protocol: A self-contained specification of the basic protocol standard for the Internet electronic mail transport. The SMTP server does not ship in Windows Vista but does ship in Windows Longhorn server. Exchange 2007 ships a SMTP stack as well.

Simple Network Management Protocol (SNMP): The Simple Network Management Protocol (SNMP v2) standard conveys management information between agents and management stations. Operations of the protocol are carried out under an administrative framework that defines authentication, authorization, access control. Management Information Base (MIB) specifications define specific values and variables that may be transferred between the SNMP agent and SNMP manager. MIBs included are SNMP framework MIB, SMI MIB, and host resource MIB.

Simple Network Time Protocol (SNTP): The Simple Network Time Protocol (SNTP) standard is an adaptation of the Network Time Protocol (NTP) used to synchronize computer clocks in the Internet. SNTP can be used when the ultimate performance of the full NTP implementation described in RFC-1305 is not needed or justified.

Simple Object Access Protocol (SOAP) 1.1: SOAP is a lightweight protocol standard for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

Simple Service Discovery Protocol (SSDP): The Simple Service Discovery Protocol (SSDP) standard provides a mechanism whereby network clients, with little or no static configuration, can discover network services. SSDP accomplishes this by providing for multicast discovery support as well as server based notification and discovery routing.

Smart Card CCID version 1.1: This protocol standard is used for communication with USB-CCID compliant smart card readers by the computer system. This protocol specifies the commands that are supported by smart card readers to facilitate the reading of smart cards.

Smart Card ICCD version 1.0: This protocol standard is used for communication with USB-ICCD compliant smart cards by the computer system. This protocol specifies the commands that are supported by integrated smart card and reader devices.

SMTP Service Extension for Authentication: This protocol standard defines an SMTP service extension whereby an SMTP client may indicate an authentication mechanism to the server, perform an authentication protocol exchange, and optionally negotiate a security layer for subsequent protocol interactions.

SMTP Service Extension for Remote Message Queue Starting: This protocol standard defines an extension to the SMTP service whereby an SMTP client and server may interact to give the server an opportunity to start the processing of its queues for messages to go to a given host. This extension is meant to be used in startup conditions as well as for mail nodes that have transient connections to their service providers.

SMTP Service Extension for Secure SMTP over Transport Layer Security: This protocol standard is an extension to the SMTP (Simple Mail Transfer Protocol) service that allows an SMTP server and client to use TLS (Transport Layer Security) to provide private, authenticated communication over the Internet. This gives SMTP agents the ability to protect some or all of their communications from eavesdroppers and attackers.

SOAP 1.1 Binding for MTOM 1.0: This specification details modifications to the SOAP Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specs necessary to successfully use these technologies with SOAP 1.1. The result is a MIME Multipart/Related XOP package: one body part, the root, containing an XML 1.0 representation of the modified SOAP 1.1 envelope, with an additional part used to contain the binary representation of each element that was optimized. The Windows SDK includes sample code to use this data structure.

SOAP Message Transmission Optimization Mechanism (MTOM): SOAP Message Transmission Optimization Mechanism (MTOM) is a SOAP message encoding mechanism designed to optimize the transmission of base64 encoded data. The protocol specifies how SOAP envelopes are serialized using XML-binary Optimized Protocol [XOP] packaging encapsulated in a Multipart/Related MIME type and carried over HTTP. The Windows SDK includes samples to invoke this protocol.

SOAP Message Transmission Optimization Mechanism (MTOM) Policy Assertion: This document defines a WS-Policy policy assertion for specifying that the SOAP Message Transmission Optimization Feature (MTOM) encoding mechanism is to be used. The policy assertion is scoped to the endpoint and specifies that all messages sent to and received from the endpoint must be optimized using MTOM. The Windows SDK includes samples to use this data structure.

SOAP Version 1.2: SOAP Version 1.2 is a lightweight protocol standard intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. SOAP 1.2 has a number of changes in syntax and provides additional (or clarified) semantics from those described in SOAP 1.1. The SOAP 1.2 specifications have been provided in two parts. [SOAP Version 1.2 Part1: Messaging Framework] provides an abstract Infoset-based definition of the SOAP message structure, a processing model and an underlying protocol binding framework, while [SOAP Version 1.2 Part2: Adjuncts] provides serialization rules for conveying that infoset as well as a particular HTTP binding.

SOAP-over-UDP: This protocol standard defines a one-way message-exchange pattern, a request-response message-exchange pattern and multicast transmission message-exchange pattern for SOAP envelopes carried in user datagrams. The Windows implementation of this profile is exercised by Windows Web Services API. There is no native usage of this implementation in Windows 7. In non-native usage, it is possible to implement both client and server roles on both Windows Client and Server SKUs.

Sockets Direct Protocol (SDP): The Sockets Direct Protocol (SDP) is a protocol standard originally defined by the Software Working Group (SWG) of the InfiniBand Trade Association (IBTA). It defines a standard wire protocol to support stream sockets (SOCK_STREAM) networking over InfiniBand. SDP utilizes various InfiniBand features (such as remote DMA (RDMA)), memory windows, solicited events etc.) for high-performance zero-copy data transfers. SDP has been adapted to work over RDMA fabrics built on top of TCP/IP-based networks by the RDMA Consortium.

Spanning Tree Protocol (STP): The network bridge functionality uses the IEEE spanning tree algorithm (STA) to establish a loop-free forwarding topology when used to interconnect two or more network interfaces. PCs and devices that support the spanning tree algorithm transmit and receive Ethernet frames as defined in IEEE 802.1D-1998 to determine presence of other bridges on the network. The network bridge functionality and STA are included in most OS SKUs but are not enabled by default.

Standard Protocol for Authentication in Host Attachments of Transient Storage Devices: The IEEE 1667 specification defines a standard protocol for secure authentication and creation of trust between a secure host and a directly attached Transient Storage Device (TSD), such as a USB flash drive, portable hard drive, or cellular phone. The protocol has only an indirect relationship with data integrity/security, and does not directly address issues of authorization and enforcement. The protocol also does not address devices that are attached using a network connection. However, a device that uses a point-to-point wireless connection such as WUSB may comply with this protocol.

Standard Protocol for Authentication in Host Attachments of Transient Storage Devices v1.1: This protocol standard consists of extensions to the IEEE 1667 1.0 specification that are being proposed for a 1.1 revision and additional extensions proposed by the working group. The additional functionality includes the password silo and updates to the authentication silo.

Subnet Bandwidth Manager (SBM) Protocol: The Subnet Bandwidth Manager (SBM) protocol standard is a signaling protocol for RSVP-based admission control over IEEE 802-style networks. SBM provides a method for mapping an Internet-level setup protocol, such as RSVP, onto IEEE 802 networks.

T.120: The T.120 protocol standard provides a means of telecommunicating all forms of Data/Telematic media between two or more multimedia terminals, and of managing such communication. The T.120 standards define the data conferencing portion of a multimedia teleconference. The recommendations specify how to efficiently and reliably distribute files and graphical information over a network. T.120 also has provision for the management of real-time (conversational) speech and video whose information signals are transmitted on channels separate from that carrying the T.120 protocol. The T.120 protocol can handle one or more simultaneous "conferences"; any terminal may participate in more than one of these if authorized to do so; the convenor of any one conference may control the participation in that conference and the information which flows in that conference. T.120 does not inherently impose constraints on the configuration of the physical connections between terminals: they may be all connected to one star-point, or connected one to two others in a chain, or a chain of star-points, and so on. Constraints may be imposed upon T.120 by the type of network in use for a specific conference. No constraint is placed on the volume of information transmitted within the various media; the T.120 protocol has the capability to organize appropriate capacity, within the constraints imposed by the type of network and connections established thereon, and taking heed of the priorities which may be set from the applications above the T.120 protocol.

T.123 Protocol: T.123 specifies an additional transport layer to allow computer networks that are not connection-oriented to make them functionally connection oriented. The T.123 protocol standard is a mandatory provision of the t.120 stack. MCS (Multipoint Communication Service) expects its underlying Transport to provide reliable point-to-point sequenced data delivery of its PDUs and to segment that data if necessary. T.123 is designed to provide open and easily extended network support for both standardized and non-standardized protocols. The basic T.123 presents a uniform OSI Transport interface and services (X.214/224) to the MCS layer above. Connection-oriented profiles are provided for switched telecom and packet switched networks.

T.124 Protocol: The T.124 standard defines the document conferencing and application sharing (sometimes called data conferencing) portion of a multimedia teleconference. The recommendations specify how to efficiently and reliably distribute files and graphical information over a network. It encompasses Generic Conference Control (GCC) functions such as conference establishment and termination, managing the roster of nodes participating in a conference, managing the roster of Application Protocol Entities and Application Capabilities within a conference, registry services for use by Application Protocol Entities, coordination of conference conductorship, as well as other miscellaneous functions.

T.126 Protocol: The T.126 standard defines a protocol supporting the management of common multi-layer visual spaces and the multipoint exchange of graphical information directed to these spaces including images, pointers, and filled and unfilled parametric drawing elements (points, lines, polygons and ellipses). In addition, keyboard and pointing device exchanges are specified to support basic user interaction. Protocol elements for creating and referencing archived visual spaces are defined to allow pre-stored or pre-distributed graphical materials to be referenced. This protocol uses services provided by Recommendations T.122 (MCS) and T.124 (GCC) and complies with the guidelines specified in Recommendation T.121 (GAT).

T.127 Protocol: The T.127 standard defines a protocol to support the interchange of binary file data within an interactive conferencing or group working environment where the T.120 suite of standards is in use. It provides mechanisms to support simultaneous distribution of multiple files, selective distribution of files to a subset of participants and retrieval of files from remote sites. T.127 uses a control/data channel architecture to facilitate simultaneous transfer of one or more binary files. It enables files to be broadcast to all participants within a conference, or to be directed selectively to a subset of sites as a private file transfer. No restrictions are placed on the type of data being transmitted. Two types of channels are used within T.127; control channels and data channels. Control channels are used for managing all aspects of the file transfer (offering files, requesting files), whereas data channels are used exclusively for the transfer of file data. Only one file can be transmitted on each data channel at a time, but additional data channels can be used to allow distribution of multiple files simultaneously. The number of data channels in use at any given time depends on the number of concurrent file transfers in progress.

T.128 Protocol: The T.128 standard defines a protocol that supports multipoint application sharing. The T.128 protocol supports multipoint computer application sharing by allowing a view onto a computer application executing at one site to be advertised within a session to other sites. Each site can, under specified conditions, take control of the shared computer application by sending remote keyboard and pointing device information. This style of application sharing does not require and does not make provision for synchronizing multiple instances of the same computer application running at multiple sites. Instead, it enables remote viewing and control of a single application instance to provide the illusion that the application is running locally.

TCP Extensions for High Performance: These extensions define TCP options for scaled windows and timestamps, to extend the domain of TCP's application to match increasing network capability. The extensions are designed to provide compatible interworking with TCP implementations that do not have the extensions. The timestamps are used for two distinct mechanisms: RTTM (Round Trip Time Measurement) and PAWS (Protect Against Wrapped Sequences).

Telnet Protocol standard: The Telnet protocol defines a standard method for interfacing terminal devices and terminal-oriented processes.

Teredo Protocol: The Teredo protocol standard enables computers located behind a Network Access Translation (NAT) to obtain an IPv6 address, which can then be used by IPv6-enabled applications (peer-to-peer applications for example) just like any other IPv6 address. This allows applications to work across NATs without requiring any changes to be NAT-aware or even Teredo-aware. That is, Teredo sits "below" IPv6 and provides a link (like Ethernet does) that supports basic IPv6 Internet connectivity. The protocol includes signaling messages, which flow between two Teredo clients as well as between Teredo clients and Teredo servers. It also includes a data channel that can flow between Teredo clients or between a Teredo client and Teredo relay. All Windows SKUs include Teredo client and relay functionality. Microsoft hosts a Teredo server as a service, and this functionality is now in Windows Server 2008 R2.

Transaction Internet Protocol: The Transaction Internet Protocol (TIP) standard requires a reliable ordered stream transport with low connection setup costs. In an Internet (IP) environment, TIP operates over TCP, optionally using TLS to provide a secured and authenticated connection, and optionally using a protocol to multiplex light-weight connections over the same TCP or TLS connection.

Transmission Control Protocol (TCP): The Transmission Control Protocol (TCP) standard is one of the core protocols of the Internet protocol suite. Using TCP, applications on networked hosts can create connections to one another, over which they can exchange data in packets. The protocol guarantees reliable and in-order delivery of data from sender to receiver. TCP also distinguishes data for multiple connections by concurrent applications (e.g. Web server and e-mail server) running on the same host. TCP is used by many of the Internet's most popular application protocols and resulting applications, including the World Wide Web and e-mail.

Transport Protocol Class 4 (TP4): Transport Protocol Class 4 (TP4) is an OSI (Open Systems Interconnect) connection-oriented protocol standard used with a Connectionless Network Protocol (CLNP). The protocol defines the interactions between peer transport entities through the exchange of transport protocol data units, and specifies mechanisms for negotiating the class of procedures to be used by the transport entities setting up a connection, and the structure and encoding of the transport protocol data units.

Trivial File Transfer Protocol (TFTP): Trivial File Transfer Protocol (TFTP) is a simple protocol standard that transfers files; it can only read or write files (or e-mail messages) to or from a remote server. **TTY Terminal Emulation**: The TTY Emulation protocol standard mimics the character set and control characters used by a Teletype-33 (TTY) serial terminal. The character set is US-ASCII and the control characters include a number of escape sequences. An escape sequence is one or more US-ASCII characters preceded by the escape character, ESC (0x1B). Further information on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_tty_emulation_protocol.asp.

Universal Description Discovery and Integration (UDDI) Version 2.04 Protocol: Universal Description, Discovery and Integration, or UDDI, is the name of a group of Web-based registries that expose information about a business or other entity and its technical interfaces (or API's). These registries are run by multiple Operator Sites, and can be used by anyone who wants to make information available about one or more businesses or entities, as well as anyone that wants to find that information. By accessing any of the public UDDI Operator Sites, anyone can search for information about Web services that are made available by or on behalf of a business. The benefit of having access to this information is to provide a mechanism that allows others to discover what technical programming interfaces are provided for interacting with a business for such purposes as electronic commerce, etc. The benefit to the individual business is increased exposure in an electronic commerce enabled world. The information that a business can register includes several kinds of simple data that help others determine the answers to the questions "who, what, where and how". Simple information about a business – information such as name, business identifiers (D&B D-U-N-S Number®, etc.), and contact information answers the question "Who?" "What?" involves classification information that includes industry codes and product classifications, as well as descriptive information about the services that the business makes available. Answering the question "Where?" involves registering information about the URL or email address (or other address) through which each type of service is accessed. Finally, the question "How?" is answered by registering references to information about interfaces and other properties of a given service. These service properties describe how a particular software package or technical interface functions. These references are called tModels in the UDDI documentation. The server role was shipped in Windows Server 2003 and Windows Server 2008 but is no longer implemented in server SKUs in Windows 7. However the UDDI SDK that implements the protocol on the client side will continue to ship in the Windows 7 SDK. Any application using UDDI SDK to talk to a UDDI registry for service metadata is a UDDI client. Currently there is no client code shipping with Windows 7.

Universal Plug and Play (UPnP) AVTransport Protocol: The Universal Plug and Play (UPnP) AVTransport Protocol standard is used by clients to control the streaming of audio/video (AV) content from a server. Media Center uses this protocol, as a client, with broadcast tuner devices that support either the Digital Receiver Interface Transceiver (DRIT) or Protected Broadcast Driver Architecture (PBDA) over IP protocol to initiate and control streaming aspects of the tuned broadcast AV stream from the tuner. This protocol is used in the Windows 7 client and by Media Center in the Windows 7 Entertainment Server SKUs.

Universal Plug and Play (UPnP) Connection Manager Protocol: The Universal Plug and Play (UPnP) Connection Manager Protocol standard is used by clients to negotiate streaming connections to a server. Media Center uses this protocol as a client with broadcast tuner devices that support either the Digital Receiver Interface Transceiver (DRIT) or Protected Broadcast Driver Architecture (PBDA) over IP protocol to create a Universal Plug and Play (UPnP) AVTransport connection. This protocol is used in the Windows 7 client and by Media Center in the Windows 7 Entertainment Server SKUs.

Universal Serial Bus (USB) Mass Storage Protocol: The USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport Specification is approved for use only with full speed floppy disk drives. CBI shall not be used in high-speed capable devices, or in devices other than floppy disk drives.

Universal Serial Bus (USB) Protocol: The Universal Serial Bus (USB) Core protocol standard is an external bus architecture for connecting USB-capable peripheral devices to a host computer. The USB

Core is maintained by the USB Implementers Forum. For information on Interface - USB Core, see http://www.usb.org/.

UPnP IGD v1.0 Device Control Protocol: This is a communication between a client on a private IP network and a server that performs Network Address Translation (NAT) between the private IP network & public IP networks (such as the Internet). Provides a way for the client to request TCP/UDP port openings through the NAT router. This protocol standard is implemented in the Windows Client SKUs and the Windows component of the Small Business Server SKU.

UPnP: Device & Service Templates: This describes two XML documents defined by the UPnP forum to describe devices and services. The UPnP Device Template is a template listing device type, required embedded devices (if any), and required services. The UPnP Service Template is a template listing action names, parameters for those actions, state variables, and properties of those state variables.

USB HID: This describes the Human Interface Device (HID) class for use with Universal Serial Bus (USB). Concepts from the USB Specification are used but not explained in this document. The HID class consists primarily of devices that are used by humans to control the operation of computer systems. Typical examples of HID class devices include:

- Keyboards and pointing devices—for example, standard mouse devices, trackballs, and joysticks.
- Front-panel controls—for example: knobs, switches, buttons, and sliders.
- Controls that might be found on devices such as telephones, VCR remote controls, games or simulation devices—for example: data gloves, throttles, steering wheels, and rudder pedals.
- Devices that may not require human interaction but provide data in a similar format to HID class devices—for example, bar-code readers, thermometers or voltmeters.

Many typical HID class devices include indicators, specialized displays, audio feedback, and force or tactile feedback. Therefore, the HID class definition includes support for various types of output directed to the end user.

USB Mass Storage Protocol: The USB Mass Storage Protocol standard, also called RBC (Reduced Block Command) is a block storage protocol, representing subset of the SCSI (Small Computer System Interface) protocol. The protocol is used to support externally or internally connected mass storage devices, like flash disks or external hard drives. The protocol is designed and used on the USB bus.

USB Modem: This specification provides information to guide implementers in using the USB logical structures for communication devices. This information applies to manufacturers of communication devices and system software developers.

User Datagram Protocol (UDP): This User Datagram Protocol (UDP) standard is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) is used as the underlying protocol. This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP).

UTF-8 Encoding: The Unicode Standard and ISO/IEC 10646 define a large character set called the Universal Character Set (UCS), which encompasses most of the world's writing systems. UTF-8 is one of several character encoding forms defined by Unicode (and ISO/IEC 10646). UTF-8 is variable length encoding scheme in which each single UCS character is represented by a sequence of one or more octets (bytes). UTF-8 uses all bits of an octet, but has the quality of reserving the full US-ASCII range: US-ASCII characters are encoded in one octet having the normal US-ASCII value, and any octet with such a value can only stand for a US-ASCII character, and nothing else. The current IETF RFC describing UTF-8, RFC 3629, clearly states that that Unicode Standard is the authoritative source for the definition of this encoding. Note that the XML 1.0 specification requires all XML processors to accept XML encoded in UTF-8. Any XML-based protocol or format should be presumed to accept UTF-8 unless that protocol or format specification contains a statement to the contrary.

Viewdata Terminal Emulation: The Viewdata Terminal Emulation protocol standard mimics the character set and control characters used by a Viewdata serial terminal. The character set is US-ASCII and the control characters include a number of escape sequences. An escape sequence is one or more US-ASCII characters preceded by the escape character, ESC (0x1B). This is typically used via a V.23-compliant modem over PSTN. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_viewdata_emulation.asp.The back-end device could be a mainframe.

VML (Vector Markup Language): VML is an application of Extensible Markup Language (XML) 1.0 which defines a format for the encoding of vector information together with additional markup to describe how that information may be displayed and edited.

VT100 Terminal Emulation: The VT100 Terminal Emulation protocol standard mimics the character set and control characters used by a family of Digital Equipment Corporation serial terminals. In the original hardware terminal, the character set is US-ASCII and the control characters include a number of escape sequences. An escape sequence is one or more US-ASCII characters preceded by the US-ASCII escape character, ESC (0x1B). Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_vt100_emulation.asp.

VT100 Terminal Emulation: Japanese Extensions (VT100J): The VT100J character set option supports the use of ASCII Double Byte Character Set (DBCS) characters in the transmitted data stream. DBCS provides support for extended language alphabets and for some East Asian language alphabets, including Chinese, Japanese, and Korean. DBCS uses the numbers 0–128 to represent the US-ASCII character set. Some numbers greater than 128 function as lead-byte characters, which are indicators that the following value is a character from a non-Latin character set. In DBCS, ASCII characters are only 1 byte in length, whereas Japanese, Korean, and other East Asian characters are 2 bytes in length. For more information on DBCS, contact the ANSI standards organization. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_vt100j_character_set_options.asp.

VT52 Terminal Emulation: The VT52 Terminal Emulation protocol standard mimics the character set and control characters used by a family of Digital Equipment Corporation serial terminals. In the original VT52 terminal, the character set is US-ASCII and the control characters include a number of escape sequences. An escape sequence is one or more US-ASCII characters preceded by the US-ASCII escape character, ESC (0x1B). Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_vt52_emulation.asp.

Web Distributed Authoring and Versioning (WebDAV) Protocol: WebDAV defines HTTP protocol extensions that enable users to collaboratively edit and manage files on remote Web servers. It provides functionality to create, change and move documents on a remote server (typically a Web server or "Web share"). WebDAV supports file locking and storage of meta-data properties such as creation date, author, and modified date. The Windows implementation of the client side of this protocol conforms to the RFC. The server portion conforms to only portions of the RFC, and thus is addressed in a separate entry.

Web Services Addressing 1.0 – Core: Web Services Addressing provides transport-neutral mechanisms to address Web services and messages. Web Services Addressing 1.0 - Core (this document) defines a set of abstract properties and an XML Infoset [XML Information Set] representation thereof to reference Web services and to facilitate end-to-end addressing of endpoints in messages. This protocol enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner. This version of the protocol is the Recommendation of the W3C Web Services Addressing Working Group, published 2006/05/09.

Web Services Addressing 1.0 - SOAP Binding: Web Services Addressing provides transport-neutral mechanisms to address Web services and messages. Web Services Addressing 1.0 - SOAP Binding defines the binding of the abstract properties defined in Web Services Addressing 1.0 - Core to SOAP Messages.

Web Services Addressing 1.0 - WSDL Binding: Web Services Addressing provides transport-neutral mechanisms to address Web services and messages. Web Services Addressing 1.0 - WSDL Binding defines how the abstract properties defined in Web Services Addressing 1.0 - Core are described using WSDL. The classes of products for which this specification is designed to be relevant include WSDL and WS-Addressing EPR consumers. The Windows SDK includes samples to exercise this protocol.

Web Services Description Language 1.1: Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. As communications protocols and message formats are standardized in the Web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

Web Services Enumeration (WS-Enumeration): WS-Enumeration is a general SOAP-based protocol standard for enumerating a sequence of XML elements that is suitable for traversing logs, message queues, or other linear information models.

Wi-Fi Protected Access (WPA): WPA provides data protection and network access control for 802.11 wireless networks. Note that this specification allows an ad hoc mode in which a client can communicate directly with another client in the absence of the server role. In Vista, this mode is not supported, and in XP this is not supported by any known drivers.

Wi-Fi Multimedia (WMM): WMM adds prioritized capabilities to Wi-Fi networks and optimizes their performance when multiple concurring applications, each with different latency and throughput requirements, compete for network resources. The specification also includes provisions for power save operation.

Wi-Fi Protected Setup (WPS) Protocol: The Wi-Fi Protected Setup Protocol standard provides wireless settings (e.g. SSID, Network Key) to an 802.11 Access Point and provides a configuration service to other 802.11 devices to supply settings for joining the wireless network.

WS-Addressing: WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

WS-Addressing Policy Assertion: This data structure defines a WS-Policy policy assertion for specifying that messages sent to and received from an endpoint must use the WS-Addressing protocol. The Windows SDK includes samples that use this data structure.

WS-AtomicTransaction Version 1.0 Protocol: This protocol standard provides the definition of the atomic transaction coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification. The specification defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, volatile two-phase commit, and durable two-phase commit. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of short-lived distributed activities that have the all-or-nothing property.

WS-Coordination Version 1.0 Protocol: This protocol standard provides support for a general coordination context. This context is needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment. Additionally this specification describes a definition of the structure of context and the requirements for propagating context between cooperating services.

WS-Discovery: The WS-Discovery protocol standard defines a flexible way to discover Web services and Web services based devices. It enables services to advertise themselves when they join and announce their departure from the network as they remove themselves from a network. Additionally it allows clients to query the network for a specific or a broad range of Web services and Web services based devices. The protocol allows for flexibility by allowing the following:

- Clients can query a network for specific services
- Clients can query a network for any service which matches a stated criteria
- Services can respond to queries directly to the client
- Services can announce their presence on a network

WS-Discovery: Managed Mode Profile: WS-Discovery v1.1 CD1 Managed Mode allows cross subnet discovery of Web services endpoints. This allows roaming clients and devices to locate services not in their immediate topological presence. The Committee Draft 1 of WS-Discovery v1.1 specification is published by the OASIS WS-Device and Discovery Technical Committee. Windows implements the Managed Mode section of WS-Discovery v1.1 CD1. Windows additionally uses the Ad-hoc section of WS-Discovery v1.0 to locate proxies which implement WS-Discovery v1.1 CD1 Managed Mode. Link to specification is as follows: http://docs.oasis-open.org/ws-dd/dpws/1.1/cd-01/wsdd-dpws-1.1-spec-cd-01.pdf.

WS-Eventing: This specifies a protocol standard for one Web Service to register interest (i.e. subscribe) with another Web Service to receive notification messages about events. The protocol defines mechanisms to register, revoke, renew, and inquire status of subscriptions.

WS-Federation: This specification defines extensions to WS-Trust and new mechanisms to facilitate federation of credentials between security realms for Web Services including usage from Web browsers. These extensions include federation metadata documents and discovery, indicating/selecting federations, reference tokens for low-bandwidth, and pseudonym management and usage. The client for the specified protocol is a Web browser, so Windows client ships the client role however there is no knowledge of the protocol specific semantics in the Windows client SKU.

WS-I Basic Profile Version 1.1: This document specifies interoperability guidance for a core set of standard Web services specifications, such as HTTP, SOAP, WSDL and UDDI, along with interoperability-promoting clarifications and amendments to those specifications. The Windows SDK includes samples that invoke this protocol.

WS-I Simple SOAP Binding 1.0: The Simple SOAP Binding Profile consists of those Basic Profile 1.0 requirements related to the serialization of the envelope and its representation in the message, incorporating any errata to date.

WS-Management: Web Services: Management is a DMTF standard for managing hardware and software resources in a network. The standard specifies a core set of Web services and usage requirements to expose a common set of operations that are central to all systems management. This comprises the abilities to:

- DISCOVER the presence of management resources and navigate between them.
- GET, PUT, CREATE, RENAME, and DELETE individual management resources, such as settings and dynamic values.
- ENUMERATE the contents of containers and collections, such as large tables and logs.
- SUBSCRIBE to events emitted by managed resources.
- EXECUTE specific management methods with strongly typed input and output parameters.

WS-MetadataExchange: This protocol standard defines how metadata can be treated as WS-Transfer resources for retrieval purposes, how metadata can be embedded in Web service endpoint references, and how Web service endpoints can optionally support a request-response interaction for the retrieval of metadata.

WS-Policy: The Web Services Policy Framework (WS-Policy) provides a general purpose model and corresponding syntax to describe the policies of a Web Service. WS-Policy defines a set of data structure elements that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities.

WS-PolicyAttachment: This data structure defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing WS-Policy metadata about the subject or the policies may be defined independently and associated through an external binding to the subject.

WS-ReliableMessaging Protocol: This protocol standard allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures. The protocol is described in a transport-independent manner allowing it to be implemented using different network technologies. To support interoperable Web services, a SOAP binding is defined. The Windows SDK includes samples that invoke this protocol.

WS-ReliableMessagingPolicy: This data structure describes a WS-Policy assertion that can be used to specify policy parameters for WS-ReliableMessaging (WS-RM). Policy parameters include InactivityTimeout, BaseRetransmissionInterval, AcknowledgementInterval, and ExponentialBackoff. The Windows SDK includes samples that use this data structure.

WS-SecureConversation: This specification defines extensions to WS-Trust to establish, renew, amend, and close Web Service security contexts. Specifically, this protocol uses the mechanisms of WS-Trust to establish end-to-end security contexts by creating a new security token (representing the context) which can then be used to secure messages with WS-Security.

WS-Security: This specification describes how to secure Web Services messages independent of transport and how to pass different credential types within messages. Specifically this describes how to sign or encrypt portions of the message for one or more recipients, and how to attach credentials (security tokens). The associated token profiles (Username, X.509, Security Assertion Markup Language (SAML), Rights Expression Language (REL), Kerberos) detail specific usages of those token types with WS-Security.

WS-SecurityPolicy: This defines extensions to WS-Policy to allow specification of security communication requirements for a Web Service. This information is used to determine if a requestor can communicate with a recipient (i.e., if it has compatible options).

WS-Transfer: This specification defines a mechanism for creating, reading, updating and deleting XML-based representations of data using the Web service infrastructure. Other than requiring the data to be serialized as XML, there are no other constraints on the data that is created, read, updated, or deleted. The .NET Framework implements the client role of the protocol. The server role of the protocol is implemented by an out of band (OOB) Windows targeted release (WTR) shipping for Windows Server 2008 R2, Windows Server 2008, and Windows Server 2003.

WS-Trust: This defines a Web Services protocol standard for obtaining, issuing, renewing, cancelling, and verifying security tokens and credentials. This protocol allows for a requestor to provide a set of credentials and options (such as expiry and usage domains) in order to obtain new credentials. The verification aspect allows the provided credentials to be verified based on service trust policies.

X.224 Protocol: X.224 defines a procedure and protocol standard that will allow transport connections for a T.120 conference to negotiate the availability of extended transport services. These services may include use of security protocols, transport protocols, levels of reliability for data transfer and support for address aliases. This negotiation is designed to provide backward compatibility with T.123 transport stacks that only support baseline transport connections. In addition to security and reliability services, this annex specifies a method for using an alias address list to establish extended transport connections. These aliases can be used for a variety of purposes, including proxy, gateway and call redirection services for T.120 communications.

XML namespaces: eXtensible Markup Language (XML) namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references.

XML Schema Definition (XSD): eXtensible Markup Language (XML) Schema specifies the XML Schema definition language, which offers facilities for describing the structure and constraining the contents of XML 1.0 documents, including those which exploit the XML Namespace facility. The schema language, which is itself represented in XML 1.0 and uses namespaces, substantially reconstructs and considerably extends the capabilities found in XML 1.0 document type definitions (DTDs).

XML-binary Optimized Packaging (XOP): This data structure defines the XML-binary Optimized Packaging (XOP) convention, a means of more efficiently serializing XML Infosets (see [XMLInfoSet]) that have certain types of content. An XOP package is created by placing a serialization of the XML Infoset inside of an extensible packaging format (such a MIME Multipart/Related, see [RFC 2387]). Then, selected portions of its content that are base64-encoded binary data are extracted and re-encoded (i.e., the data is decoded from base64) and placed into the package. The locations of those selected portions are marked in the XML with a special element that links to the packaged data using URIs. The Windows SDK includes samples that use this data structure.

XML-Encryption Syntax and Processing: This document specifies a process for encrypting data and representing the result in XML. The data may be arbitrary data (including an XML document), an XML element, or XML element content. The result of encrypting data is an XML Encryption

EncryptedData element which contains (via one of its children's content) or identifies (via a URI reference) the cipher data. The .NET Framework receives and emits the data structure. The data structure is received and emitted by an out of band (OOB) Windows targeted release (WTR) shipping for Windows Server 2008 R2, Windows Server 2008, and Windows Server 2003. This data structure is used in the Security Assertion Markup Language (SAML) V2.0 protocol, the WS-Federation: Browser Extensions Version 2 protocol, and the WS-Trust protocol.

XML-Signature Syntax and Processing: This specification specifies Extensible Markup Language (XML) digital signature validation rules and syntax. XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere. The roles specified in this form refer to creators (client) and validators (server) of the data structure. **XModem File Transfer Protocol:** The Xmodem File Transfer protocol standard provides simple serial file transfer between a server and client across a point-to-point link using fixed-length packets. Each server packet contains 128 bytes of file data and is individually acknowledged by the receiving client. Only one file can be sent per transmission, and the transmission must be restarted from the beginning if it fails. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_xmodem_file_transfer.asp.

XModem File Transfer Protocol: 1K Extension: The 1K extension to the Xmodem File Transfer protocol standard allows larger data blocks (1024 bytes of file data). The server uses an alternate start of header (SOH) character (0x02) to mark the start of a 1K Xmodem packet. Further information on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_1k_xmodem_file_transfer.asp

YModem File Transfer Protocol: The Ymodem File Transfer protocol standard provides simple serial file transfer between a server and client across a point-to-point link using fixed-length packets. Each server packet contains either 128 bytes (the start of header character (SOH) = 0x01) or 1024 bytes (SOH = 0x02) of file data and is individually acknowledged by the receiving client. Multiple files can be sent per transmission, and the transmission must be restarted from the beginning if it fails. Packet formats are similar to those for the HyperTerminal Xmodem File Transfer protocol. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_ymodem_file_transfer.asp.

YModem File Transfer Protocol: G Extension: The Ymodem-G File Transfer extension of the Ymodem File Transfer protocol standard can be requested by the client by polling the server with an ASCII G character instead of the ASCII C character used for the Ymodem File Transfer protocol. For the Ymodem-G File Transfer extension, the server sends data packets but does not wait for a client acknowledgement (ACK) for each individual packet. The server expects a client ACK in response to the end-of-transmission (EOT) character. If the receiver detects an error in a packet, it aborts the entire transmission by sending a cancel sequence consisting of multiple (typically 10) CAN line control characters (0x18). Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_ymodem-g_file_transfer.asp.

ZModem File Transfer Protocol: The Zmodem File Transfer protocol standard provides a simple means of point-to-point serial file transfer. The session initialization dialog includes attributes that allow an interrupted file transfer to be resumed. This protocol allows the following file transfers: Multiple files may be transferred per session. File data is transmitted in 1024-byte data blocks. Data packets are strung together and acknowledged, only once, at end-of-file. Error detection supports checksum error checking. An interrupted file transfer can be restarted from the point of interruption. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_zmodem_file_transfer.asp.

ZModem File Transfer Protocol: Crash Recovery Extension: The Zmodem with Crash Recovery protocol standard provides a simple means of point-to-point serial file transfer. The session initialization dialog includes attributes that allow an interrupted file transfer to be resumed. Multiple files may be transferred per session. File data is transmitted in 1024-byte data blocks. Data packets are strung together and acknowledged only once, at end-of-file. Error detection supports CRC error checking. Further information is on the MSDN website at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/hyperterminal_zmodem_with_crash_recovery.asp.

10.2 Standards Extended

Windows extends the following standards:

Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG): The Secret Key Transaction Authentication for DNS (TSIG) protocol standard provides transaction level authentication for DNS. TSIG is extensible through the definition of new algorithms. This protocol specifies an algorithm based on the Generic Security Service Application Program Interface (GSS-API) (RFC2743).

Internet IP Security (IPsec) Domain of Interpretation for ISAKMP: The Internet Security Association and Key Management Protocol (ISAKMP) standard defines a framework for security association management and cryptographic key establishment for the Internet. This framework consists of defined exchanges, payloads, and processing guidelines that occur within a given Domain of Interpretation (DOI). This document defines the Internet IP Security DOI (IPSEC DOI), which instantiates ISAKMP for use with IP when IP uses ISAKMP to negotiate security associations.

IrDA Object Exchange (OBEX) Protocol: The IrOBEX protocol standard (for IrDA Object Exchange, OBEX for short) is a binary protocol that enables transfer of opaque data objects via the Infrared link. A major use of IrOBEX is a "Push" and/or "Pull" of opaque data objects, meaning the local system can "push" or "pull" data from the other system via the Infrared link.

Layer 2 Tunneling Protocol (L2TP) with IPsec: This protocol standard describes the Layer 2 Tunneling Protocol (L2TP) and the utilization of IPsec for tunnel authentication, privacy protection, integrity checking and replay protection. L2TP facilitates tunneling of PPP packets across an intervening network in a way that is as transparent as possible to both end-users and applications.

Link Local Multicast Name Resolution (LLMNR) Protocol: The goal of Link-Local Multicast Name Resolution (LLMNR) is to enable name resolution in scenarios in which conventional DNS name resolution is not possible. LLMNR supports all current and future DNS formats, types, and classes, while operating on a separate port from DNS, and with a distinct resolver cache.

Point-to-Point Tunneling Protocol (PPTP): This protocol standard allows the Point to Point Protocol (PPP) to be tunneled through an IP network. PPTP does not specify any changes to the PPP protocol but rather describes a new vehicle for carrying PPP. A client-server architecture is defined in order to decouple functions which exist in current Network Access Servers (NAS) and support Virtual Private Networks (VPNs). The PPTP Network Server (PNS) is envisioned to run on a general purpose operating system while the client, referred to as a PPTP Access Concentrator (PAC) operates on a dial access platform. PPTP specifies a call-control and management protocol which allows the server to control access for dial-in circuit switched calls originating from a PSTN or ISDN or to initiate outbound circuit-switched connections. PPTP uses an enhanced GRE (Generic Routing Encapsulation) mechanism to provide a flow- and congestion-controlled encapsulated datagram service for carrying PPP packets. PPTP defines four roles:

- A client acting as PNS
- A server acting as PAC
- A proxy such as access concentrator acting as a PAC
- A server acting as a PNS

Windows client and server SKUs implement the first and second roles.

Transport Layer Security (TLS): The Transport Layer Security (TLS) protocol standard supports mutual authentication based on the client's and server's X.509 certificates and provides for subsequent confidentiality and integrity of the client/server communication channel via a session key that is derived from a master secret. Exceptions from the TLS standards will be defined in the TLS profile.