[MS-SYS]:
Windows System Overview

The Microsoft Work Group Server Protocol Program (WSPP) makes available, under license, the communications protocols that are implemented in Windows Server products and that are used to deliver file, print, and user and group administration services. This document provides licensees of the Work Group Server Protocol Program (WSPP) with an overview for the collections of licensed protocols under the WSPP program.

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

Copyright © 2011 Microsoft Corporation.

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

Revision Summary

Date	Revision History	Revision Class	Comments
03/14/2007	1.0		Version 1.0 release
04/10/2007	1.1		Version 1.1 release
05/18/2007	1.2		Version 1.2 release
06/08/2007	1.2.1	Editorial	Revised and edited the technical content.
07/10/2007	1.2.2	Editorial	Revised and edited the technical content.
08/17/2007	1.2.3	Editorial	Revised and edited the technical content.
09/21/2007	1.2.4	Editorial	Revised and edited the technical content.
10/26/2007	1.2.5	Editorial	Revised and edited the technical content.
01/25/2008	1.2.6	Editorial	Revised and edited the technical content.
03/14/2008	1.2.7	Editorial	Revised and edited the technical content.
06/20/2008	1.2.8	Editorial	Revised and edited the technical content.
07/25/2008	1.2.9	Editorial	Revised and edited the technical content.
08/29/2008	1.2.10	Editorial	Revised and edited the technical content.
10/24/2008	1.2.11	Editorial	Revised and edited the technical content.
12/05/2008	1.2.12	Editorial	Revised and edited the technical content.
01/16/2009	1.2.13	Editorial	Revised and edited the technical content.
02/27/2009	1.2.14	Editorial	Revised and edited the technical content.
04/10/2009	1.2.15	Editorial	Revised and edited the technical content.
05/22/2009	1.2.16	Editorial	Revised and edited the technical content.
07/02/2009	1.2.17	Editorial	Revised and edited the technical content.
08/14/2009	1.2.18	Editorial	Revised and edited the technical content.
09/25/2009	1.2.19	Editorial	Revised and edited the technical content.

Copyright © 2011 Microsoft Corporation.

Date	Revision History	Revision Class	Comments
11/06/2009	2.0	Major	Updated and revised the technical content.
12/18/2009	2.0.1	Editorial	Revised and edited the technical content.
01/29/2010	2.0.2	Editorial	Revised and edited the technical content.
03/12/2010	2.1	Minor	Updated the technical content.
04/23/2010	2.1.1	Editorial	Revised and edited the technical content.
06/04/2010	2.1.2	Editorial	Revised and edited the technical content.
07/16/2010	2.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	2.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	2.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	2.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	2.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	2.1.2	No change	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2011 Microsoft Corporation.

Contents

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

1 Introduction

The Work Group Server Protocol Program (WSPP) protocols can be implemented in a range of server applications to communicate or interoperate with Microsoft Windows® Server operating systems for file, print, and user and group administration applications. These collections of protocols that can be licensed together are called WSPP Tasks. This document describes the technical relationships among the set of protocols that are included in a particular WSPP Task and provides basic protocol concepts.

1.1 Overview

There are three Tasks in WSPP:

- File and Print Task
- User and Group Administration Task
- Networking and Transports Task

Each WSPP Task contains several scenarios. Each scenario comprises a number of protocols, both those within WSPP and those that are existing standards.

For each scenario, an overview diagram is presented that broadly specifies how the protocols in a scenario are related. Some scenarios comprise a significant number of protocols; for those scenarios, additional diagrams with more detail follow. Some scenarios involve only one WSPP protocol, and these, therefore, do not have additional diagrams.

There are two main ways to visualize the relationship among the protocols. The first is a strict protocol stack view, which allows the reader to determine the encapsulation of each protocol as it moves down the stack for eventual transmission on the underlying physical medium. For purposes of this document, this first view is termed the protocol stack.

The second view is an interrelational view of the protocols—a view that allows readers to determine which protocols are used in parallel with the "main" protocol. This view is useful in cases where other protocols are required for the main protocol to function; however, these other protocols may be logically contained in the main protocol or invoked separately by an implementation of the protocol. This second view is termed the logical dependencies diagram.

Copyright © 2011 Microsoft Corporation.

2 File and Print Task

The first WSPP Task, File and Print services, has five scenarios; the Basic File Services task is the most complicated and the Internet Print task is the simplest.

2.1 Basic File Services

The Basic File Services scenario comprises the remote file access protocols and a number of supporting or related protocols, as shown in the following diagram.

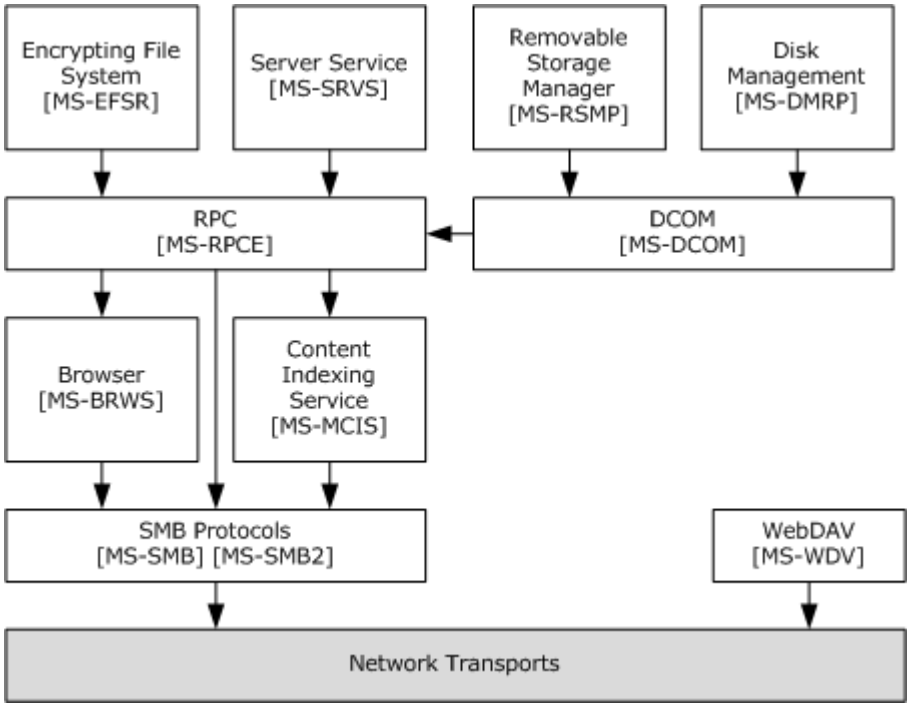


Figure 1: Remote file access protocols

The Server Message Block (SMB) protocols refer to the original Server Message Block (SMB) Protocol and its new extension, the Server Message Block (SMB) Version 2 Protocol.

These protocols serve a special role in the protocols that are available under WSPP because they offer a construct known as the named pipe. A named pipe is a logical connection, similar to a Transmission Control Protocol (TCP) session, between the client and server that are involved in the SMB connection. 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.

Many protocols in WSPP are layered on top of named pipes, either directly or indirectly, through remote procedure calls (RPCs), as specified in section 3.12. When named pipes are used, they have the advantage of insulating the higher-layer protocol from the transport that was chosen and also of offering the higher-layer protocol the authentication services of the SMB connection.

2.1.1 Protocol Stack

The client for a Server Message Block (SMB) connection can connect to the server over a variety of transports. After the client is connected, it may make a number of related requests. The following

Copyright © 2011 Microsoft Corporation.

diagram displays the protocol stack that is used by SMB. The first diagram is the lower part of the protocol stack and shows how SMB interacts with the underlying network transports.

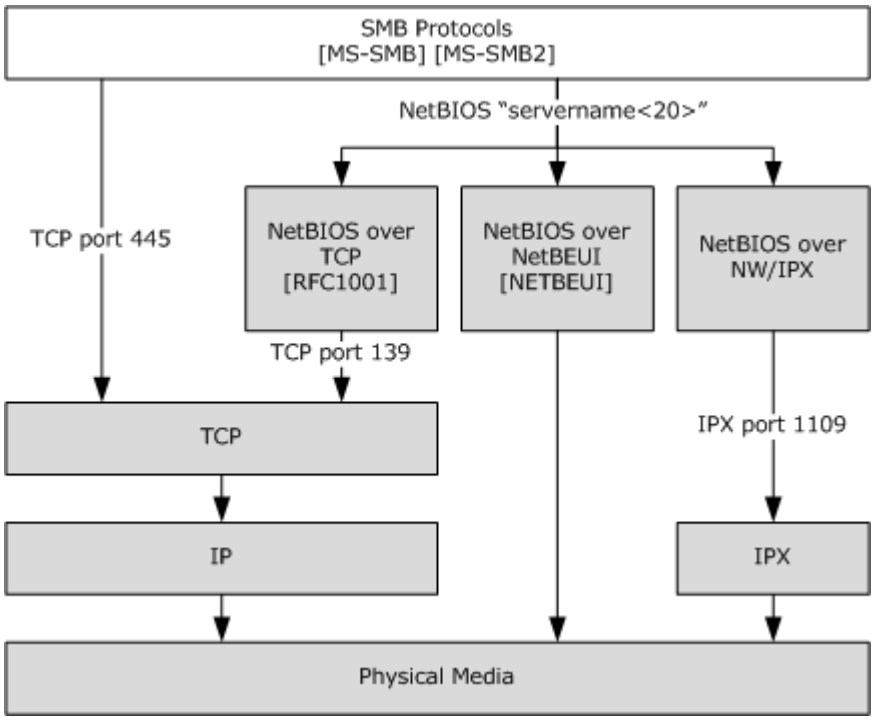


Figure 2: Protocol stack used by SMB

The SMB protocols can either use TCP natively (as specified in [RFC793]), through port 445, or they can use a NetBIOS transport. NetBIOS, in turn, is implemented as NetBIOS over another transport, either TCP (as specified in [RFC1001]), NetBEUI, or NetWare/IPX. Those underlying layers are responsible for the actual transmission over the physical medium.

The following diagram is the higher part of the protocol stack for this scenario. This diagram includes those protocols that rely on the SMB protocols for their transport, and also includes other protocols in the scenario.

Copyright © 2011 Microsoft Corporation.

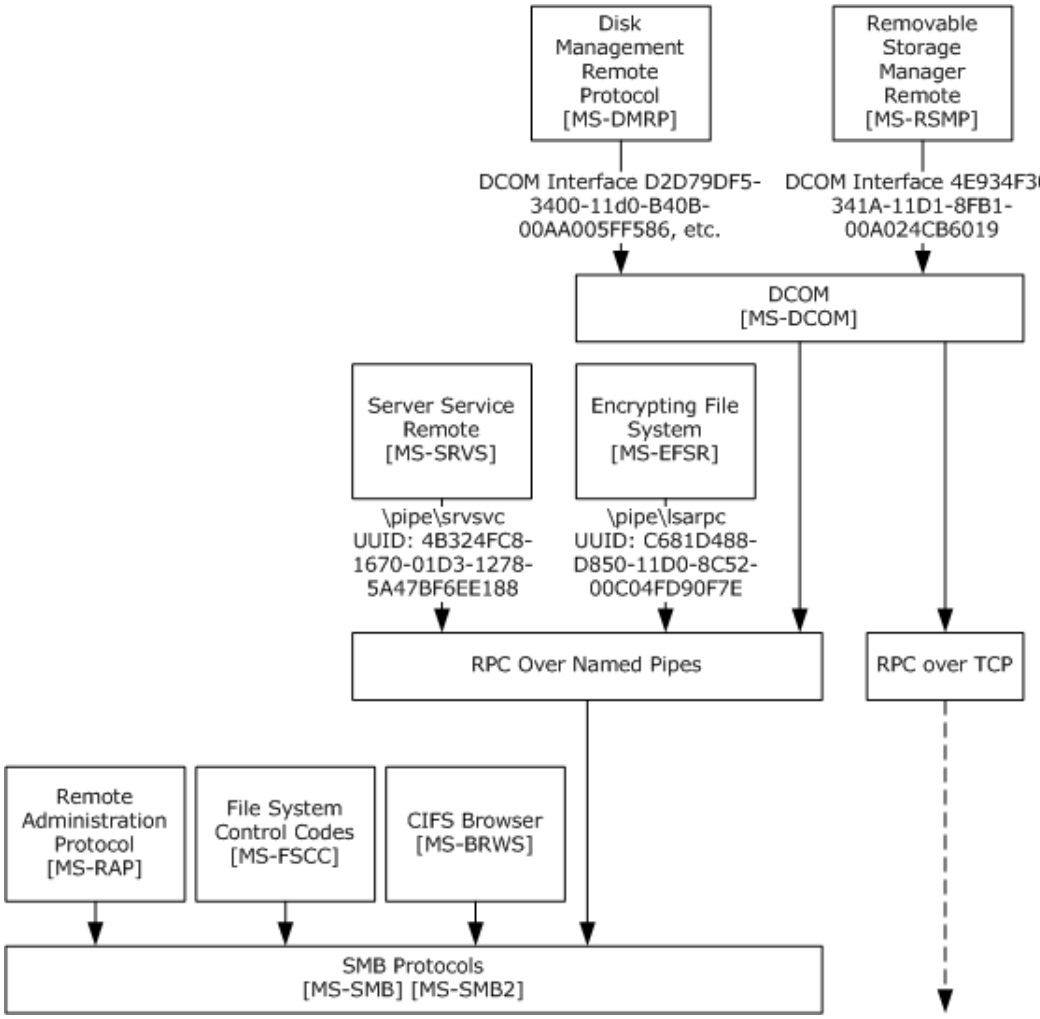


Figure 3: Protocols in the scenario

Above the SMB protocols block are protocols that extend the functionality of the file server beyond that of simple file sharing. The File System Control Codes are composed of the structured commands that can be sent to a device or file system that has been accessed through the SMB protocols. These are layered on top of the SMB protocols because they cannot be used or even expressed outside of an SMB connection.

The Server Service Remote Protocol (SRVSVC) is used to query the server for additional information, such as the named share points that can be used for connections and similar administrative information.

The Remote Administration Protocol (RAP) is used by legacy clients (MS-DOS, Microsoft Windows® 95 operating system, Microsoft Windows® 98 operating system, and Microsoft Windows® Millennium Edition operating system) for similar purposes to the Server Service Remote Protocol.

Microsoft Windows NT® operating system, Microsoft Windows® 2000 operating system, Windows® XP operating system, Windows Vista® operating system, Windows Server® 2003 operating system, Windows Server® 2008 operating system, Windows® 7 operating system, and

Copyright © 2011 Microsoft Corporation.

Windows Server® 2008 R2 operating system only use the Remote Administration Protocol in support of clients that use the Common Internet File System (CIFS) Browser Protocol for discovering computers on the local network. The Remote Administration Protocol is used by the File Access Services to transport just the NetServerEnum2 and NetServerEnum3 requests; these requests are used for obtaining a list of servers from the Browser service. The Remote Administration Protocol is expressed in the SMB protocols through the Transact command.

The Server Service Remote Protocol interface, however, is an RPC interface. The Server Service Remote Protocol interface is only accessible over an RPC named pipes binding. Named pipes are an abstraction that is offered by SMB. The RPC layer performs the necessary encoding and decoding of the commands in the Server Service Remote Protocol interface.

Similarly, the Encrypting File System Remote (EFSRPC) Protocol interface also layers on top of a named pipe endpoint, and it uses the RPC runtime for encoding and decoding.

2.1.2 Logical Dependencies

There are other protocols that are used to implement a file server that do not appear in neat layers, as do the transports and SMB higher-layer protocols. The following diagram outlines the authentication and security protocols that are used by the SMB protocols to authenticate a client. The security protocols are not layered; instead, they are contained in version 1.0 of the SMB Protocol, as specified in [MS-SMB] sections 2.2.3.1 and 3.2.4.2.4.

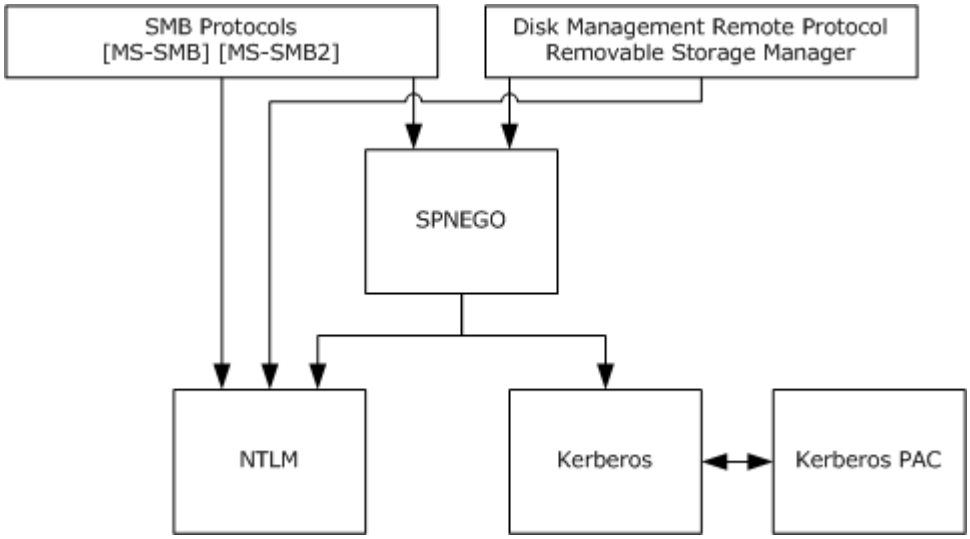


Figure 4: Authentication and security protocols

As is shown in the preceding diagram, the SMB protocols can use the Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO) Protocol Extensions for their authentication scheme, or they can use NTLM directly. In the specifications for the SMB protocols, the use of extended security for the SPNEGO Protocol Extensions or legacy NTLM is specified in [MS-SMB] section 3.2.4.2.4. For purposes of this document, it is sufficient to note that NTLM is used directly by SMB, primarily for supporting legacy clients. Note also that in this mode, the NTLM communication is not exchanged through NTLM messages, but that the communication is decomposed so that only the challenge-response fields are sent. This is specified in [MS-NLMP] section 4.

Copyright © 2011 Microsoft Corporation.

When using extended security, the SMB protocols use the SPNEGO Protocol Extensions, as specified in [MS-SPNG], to select an authentication protocol. For extended security, the Kerberos protocol is typically chosen, although in some scenarios, a downgrade to NTLM is acceptable. When the Kerberos protocol is used, authentication takes place according to normal Kerberos, but in most cases, a privilege attribute certificate (PAC) data structure is included. The PAC structure in a Microsoft Windows® domain environment contains group membership information that can be used by the file server to control access to its resources. The interaction of authentication is described in more detail in the Base Authentication and Authorization section.

Note Although SMB does not encode packets by using the Kerberos protocol, it does require a Kerberos implementation to generate the Kerberos KRB_AP_REQ message (as specified in [RFC4120] section 3.2, and in [RFC1964]) on the SMB client, and a Kerberos implementation on the CIFS/SMB server to be able to process the KRB_AP_REQUEST message according to the Kerberos protocol. Likewise, it requires an NTLM implementation to perform the authentication when NTLM is selected for authentication.

The DCOM-based protocols for disk management [MS-DMRP] and removable storage [MS-RSMP] both specify NTLM for authentication in the DCOM/RPC framework, as well as for the SPNEGO Protocol Extensions.

2.2 Distributed File System and File Replication Service

This section consists of two separate sections, the Distributed File System and the File Replication Service.

2.2.1 Distributed File System

The Distributed File System (DFS) is the means by which an arbitrarily large group of servers can be woven into a single hierarchical tree of servers, which are then presented to the user as a single large directory tree. The DFS client-to-server referral protocol runs over SMB as a special SMB command. The DFS client can inquire of the server what portion of the overall DFS namespace is covered by the server, and what portions are actually pointers to other servers. The client can then choose the server with which to communicate by examining the name of the object it is attempting to access.

The DFS server-to-server protocol is a management interface, much like the Distributed File System: Replication Helper (DFS-R Helper) Protocol, which allows the caller to manage an implementation of DFS.

Copyright © 2011 Microsoft Corporation.

2.2.1.1 Protocol Stack

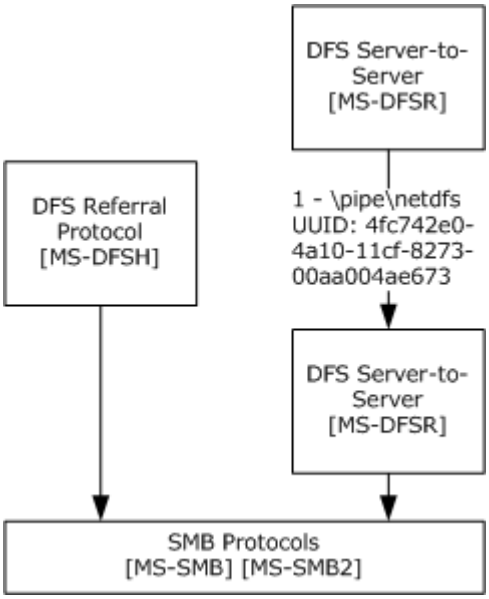


Figure 5: Protocol stack

2.2.2 File Replication Service

The File Replication Service (FRS) is the predecessor to DFS-Replication, which was used for replicating files among Microsoft Windows®-based servers. The File Replication Service (FRS) Protocol is an RPC-based protocol that uses the standard authentication services as specified in section 3.1.

File replication is performed by the servers that are participating in replication by using the FRS Protocol to communicate, and to determine which files need to be copied or deleted. After the files to be copied have been determined, they are then copied by means outside of the FRS Protocol. The Windows implementation uses SMB to perform the actual file manipulations; an alternate implementation can use other file sharing protocols.

2.2.2.1 Protocol Stack

The protocol stack for the File Replication Service (FRS) Protocol is very simple because it is just an RPC interface.

Copyright © 2011 Microsoft Corporation.

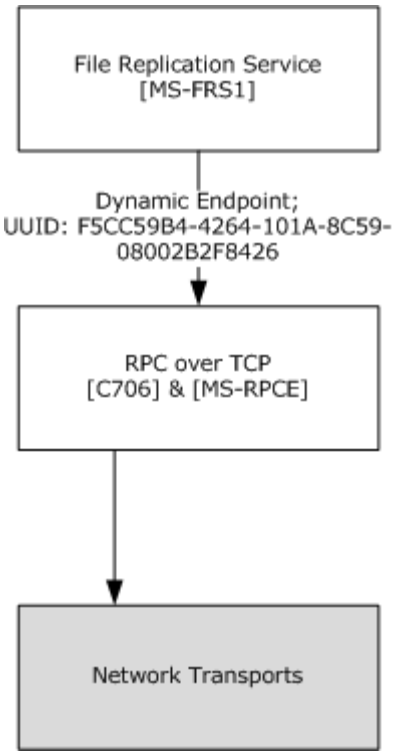


Figure 6: Protocol stack for the File Replication Service

The FRS Protocol, as specified in [MS-FRS1], uses the dynamic endpoint capability of the remote procedure call (RPC) and allows RPC to select the actual TCP port that is used.

2.3 Print RPC

The Print RPC scenario is composed of three protocols:

- The Print System Remote Protocol, as specified in [MS-RPRN].
- The Print System Asynchronous Remote Protocol, as specified in [MS-PAR].
- The Print System Asynchronous Notification Protocol, as specified in [MS-PAN].

These protocols provide mechanisms for gaining access to, and for managing print system functionality, in a distributed environment.

The Print System Remote Protocol is an RPC interface that supports synchronous access to, and management of, print system functionality.

The Print System Asynchronous Remote Protocol and the Print System Asynchronous Notification Protocol are a set of RPC interfaces that are closely modeled after the Print System Remote Protocol. This set of RPC interfaces provides asynchronous access to, and management of, print system functionality.

Copyright © 2011 Microsoft Corporation.

2.3.1 Protocol Stack

All protocols in this scenario are composed of RPC interfaces. The following diagram illustrates the resulting layers of their combined protocol stack.

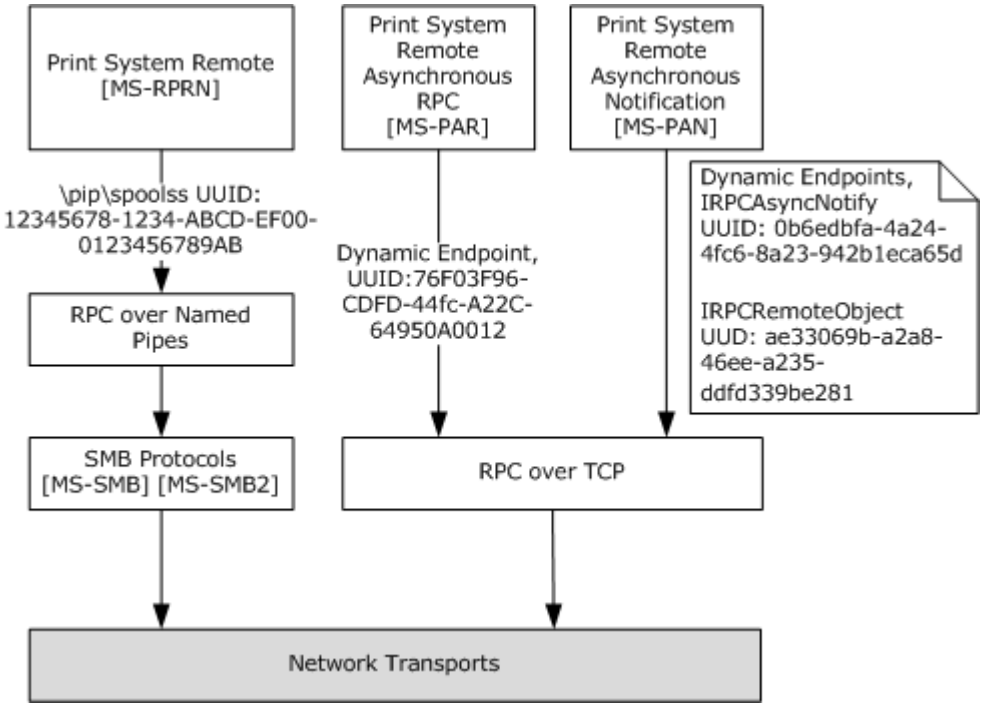


Figure 7: Combined protocol stack

For more information about the layering of RPC over named pipes and RPC over TCP transports, see the Remote Procedure Calls scenario, as specified in section 3.12, or the Networking and Transports Task, as specified in section 4.

2.3.2 Logical Dependencies

The protocols in the Print RPC scenario do not depend on any protocols other than their underlying transports. Consequently, both of their underlying transports have dependencies resulting from the authentication schemes that they support, as specified in section 3.1.

2.4 Internet Print

The Internet Print scenario consists of the Web Point-and-Print Protocol, which provides a mechanism for clients to obtain printer driver software from servers by using the Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS).

2.4.1 Protocol Stack

The following diagram illustrates the layering of the protocol in this scenario with other protocols in its stack. The Web Point-and-Print Protocol is layered directly over HTTP or HTTPS.

Copyright © 2011 Microsoft Corporation.

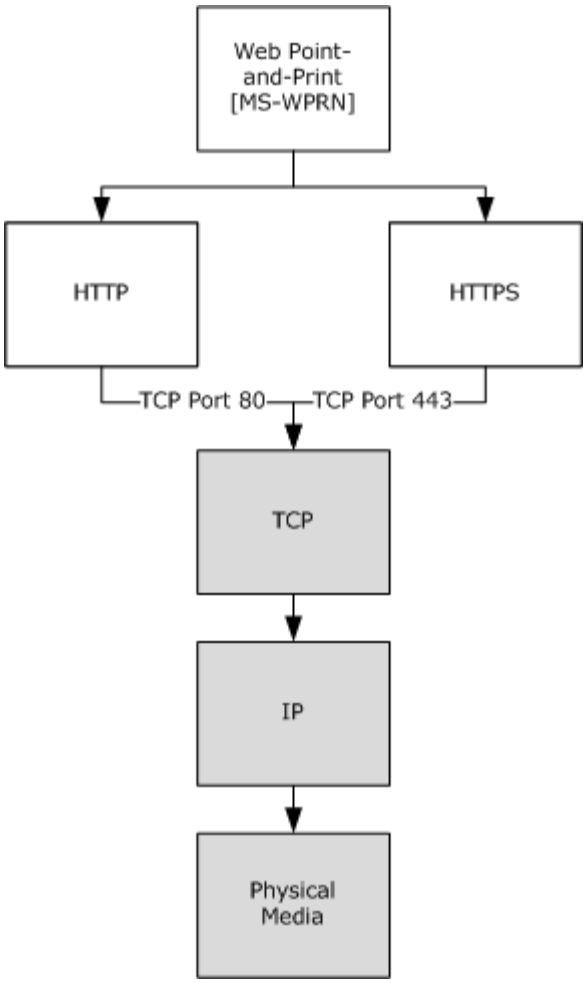


Figure 8: Protocol layering

2.4.2 Logical Dependencies

The Internet Print scenario has no logical dependencies other than its underlying transports.

2.5 Advanced File Services

The Advanced File Services scenario is composed of the protocols that allow a server to participate in a replicated distributed file system or to use the Server Message Block (SMB) Version 2 Protocol. Architecturally, version 2 of this protocol is the same as the Server Message Block (SMB) Protocol because it relies on underlying transport protocols and it exposes constructs such as named pipes to higher-layer protocols. The section covering SMB is sufficient to provide an overview of the SMB Version 2 protocol.

2.5.1 Protocol Stack

The Distributed File System and File Replication Service scenarios span several protocols and data formats. The following figure shows the overall view.

Copyright © 2011 Microsoft Corporation.

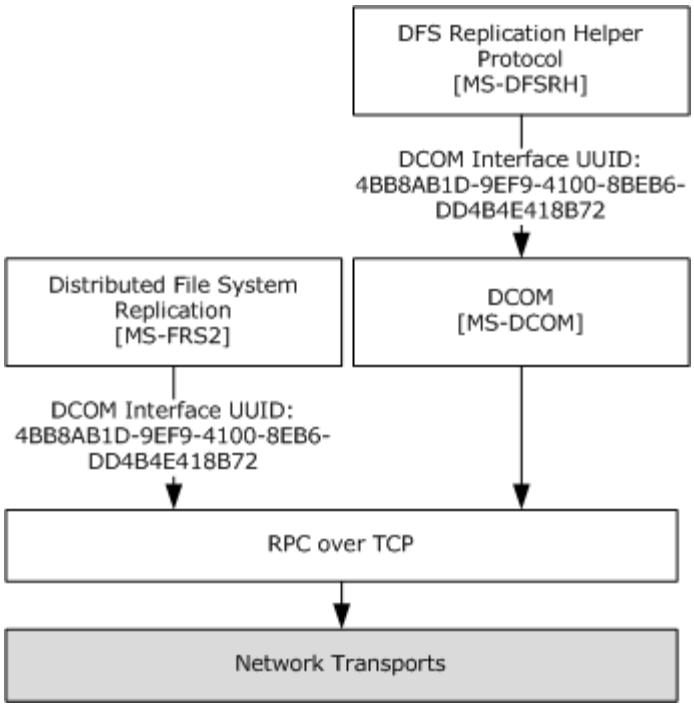


Figure 9: Distributed File System Replication

The Distributed File System and File Replication Service are also discussed in the Remote Procedure Calls scenario. They only work with RPC over TCP. The protocol is used to allow two or more servers to replicate files among them by using the remote differential compression (RDC) algorithm. The Distributed File System: Replication Helper (DFS-R Helper) Protocol is a DCOM interface that allows a client to manage a server that is participating in a DFS-replication relationship with other servers. As with the other services, the DFS system relies on the authentication services that are provided by the Kerberos protocol.

2.5.2 Logical Dependencies

Logical dependencies are represented as follows.

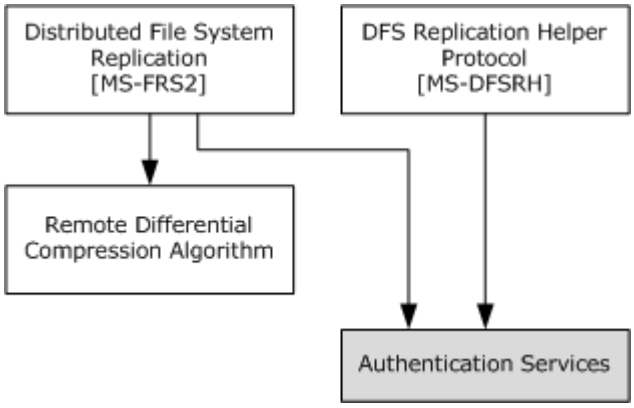


Figure 10: Logical dependencies

Copyright © 2011 Microsoft Corporation.

In the preceding diagram, the Distributed File System Replication (DFS-R) Protocol is shown to depend on the remote differential compression algorithm to accomplish its overall tasks. DFS-Replication and the DFS-R Helper Protocol are independent but are typically exposed by the same implementation, because the DFS-R Helper Protocol is used to manage an implementation of DFS-Replication. Both protocols rely on the standard authentication services.

Copyright © 2011 Microsoft Corporation.

3 User and Group Administration Task

3.1 Base Authentication and Authorization

Authentication is central to the behavior of Microsoft Windows®-based clients and servers in a domain environment, and the User and Group Administration Task reflects this. Authentication and administration of accounts are related WSPP Tasks but involve different collections of protocols. Authentication scenarios can make use of over 40 protocols. Of these, five protocols are best discussed in the context of domain interaction.

These five domain-related protocols are:

- Local Security Authority (Domain Policy) Remote Protocol
- Local Security Authority (Translation Methods) Remote Protocol
- Netlogon Remote Protocol
- Security Account Manager (SAM) Remote Protocol (Client-to-Server)
- Security Account Manager (SAM) Remote Protocol (Server-to-Server)

These protocols are discussed further in section 3.2. The Network Time Services scenario comprises three protocols for time synchronization use, as specified in section 3.10, and the Remote Procedure Calls scenario comprises three protocols for secure RPC, as specified in section 3.12.

If the previously mentioned protocols are omitted, the diagram is much easier to read.

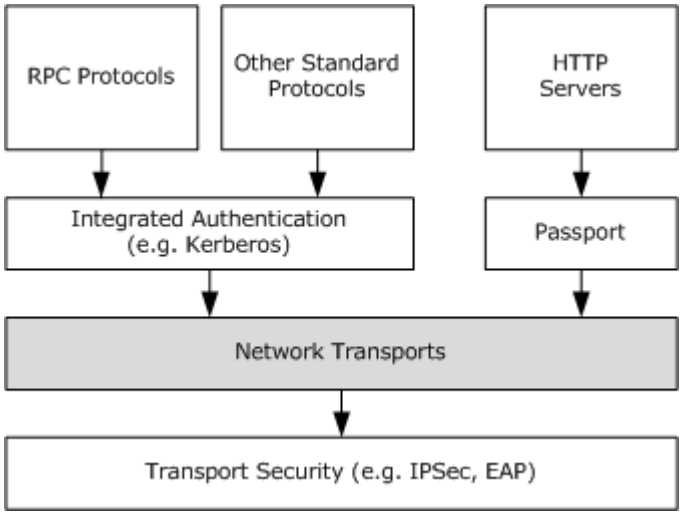


Figure 11: Protocol authentication

3.1.1 Integrated Authentication Protocols

To provide easier visualization of authentication protocols, this topic separates the involved protocols into client and server views. The following diagram shows the client view of Integrated authentication.

Copyright © 2011 Microsoft Corporation.

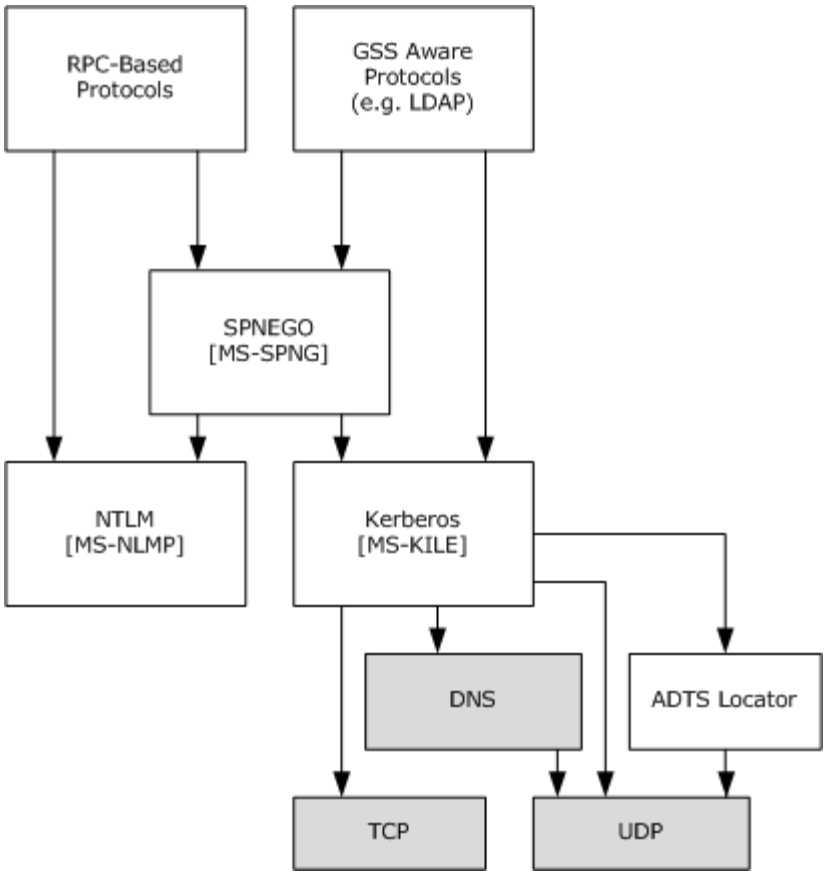


Figure 12: Client view of Integrated authentication

In the client view of authentication, RPC-based protocols use either the SPNEGO Protocol Extensions or NTLM, as specified in [NTLM], to authenticate to their servers. The SPNEGO Protocol Extensions allow for the selection of the Kerberos protocol or NTLM during authentication. Similarly, Generic Security Service (GSS)–aware protocols directly use either the SPNEGO Protocol Extensions or the Kerberos protocol; for example, in the case of the Lightweight Directory Access Protocol (LDAP), as specified in [RFC2251].

On the server side of the authentication protocols, the components are almost the same; however, interaction with the domain controller is introduced. The following diagram shows the domain controller interaction.

Copyright © 2011 Microsoft Corporation.

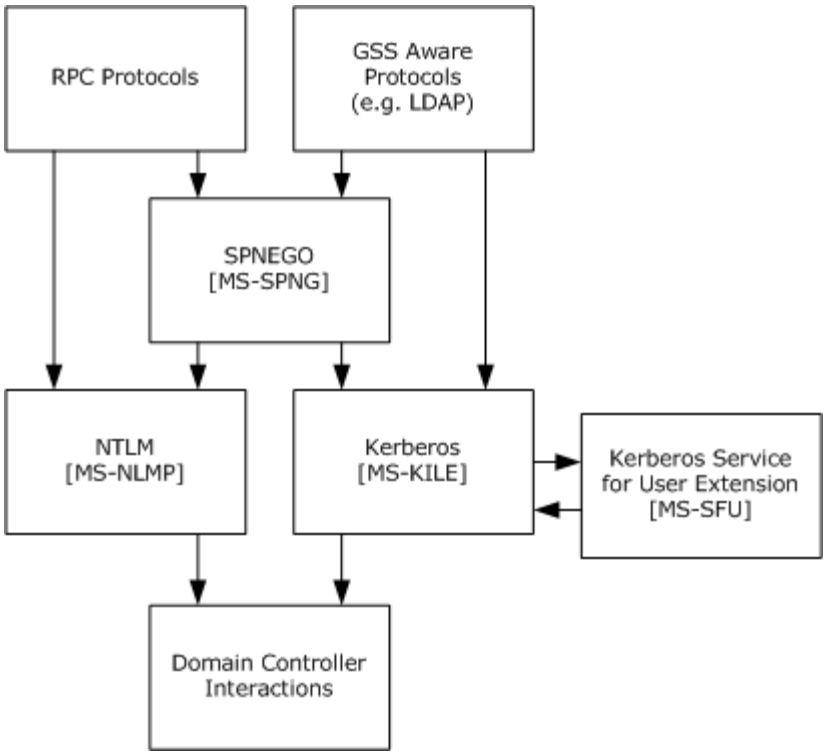


Figure 13: Domain controller interaction of the server view

Here, both NTLM and the Kerberos protocol can be seen interacting with the domain controller (for more information about the details of that interaction, see section 3.2). Also, the Kerberos protocol may invoke itself for user extensions, as specified in the Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol [MS-SFU].

3.1.2 Passport

Passport is an authentication service that is used by Microsoft MSN and its partners to authenticate users who are accessing services that require an authenticated user.

The Passport Server Side Include (SSI) Version 1.4 Protocol is used by online services to authenticate users (clients) by using the Passport authentication service. The protocol is also known as the SSI 1.4 Protocol.

The SSI 1.4 Protocol is an HTTP-based protocol (as specified in [RFC2616]) that is used to authenticate a client to a partner server, with the assistance of an authentication server. This protocol relies on HTTP cookies that are carried by standard HTTP messages to convey authentication information between an authentication server and a partner server, as specified in [MS-PASS].

3.1.2.1 Protocol Stack

The SSI 1.4 Protocol provides a simple protocol stack for Passport authentication. For example, an application that needs to use a service is asked to authenticate itself (the user). The application uses an API to the SSI 1.4 Protocol, which uses HTTP or HTTPS as its transport. HTTP(S), in turn, uses TCP at ports 80 or 443 over IP.

Copyright © 2011 Microsoft Corporation.

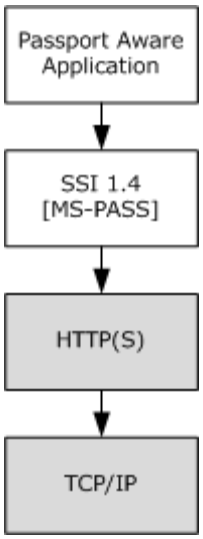


Figure 14: Simple protocol stack

3.1.2.2 Logical Dependencies

As specified in the SSI 1.4 Protocol specification, the protocol works with a client, a partner service (the service being accessed), and an authentication service (Passport). These dependencies can be represented as follows.

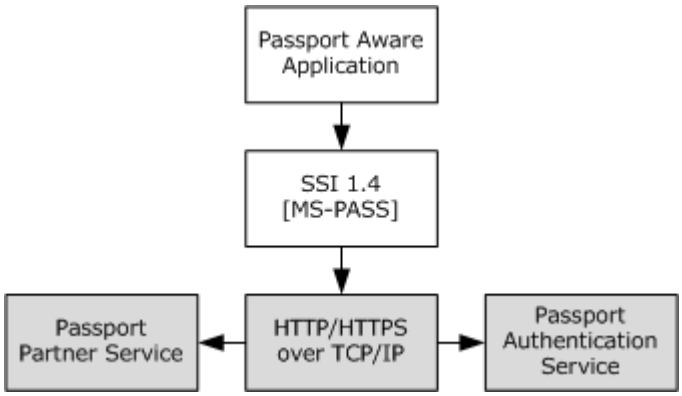


Figure 15: Protocol dependencies

During the authentication process, when the service is asked for access to a protected resource, it redirects the client to the Passport service; the Passport service prompts the client for credentials, which are submitted to Passport. The Passport service then redirects the client to the partner service and includes the successful or failed results of the authentication.

3.1.3 Transport Security (IPsec and EAP)

3.1.3.1 Internet Protocol Security Protocols Extensions

The Internet Protocol Security Protocols Extensions scenario is composed of the following elements:

Copyright © 2011 Microsoft Corporation.

- A set of extensions, as shown in the Internet Key Exchange (IKE) Protocol Extensions, to the Internet Key Exchange (IKE).
- The Authenticated Internet Protocol, which peers may use instead of the IKE Protocol Extensions when they are negotiating Internet Protocol security (IPsec) parameters.

3.1.3.1.1 Protocol Stack

The IPsec Extensions scenario retains the protocol stack as specified in [RFC2409] for the Internet Key Exchange (IKE) and specified in [RFC3947] for negotiation of network address translation (NAT)-traversal in IKE.

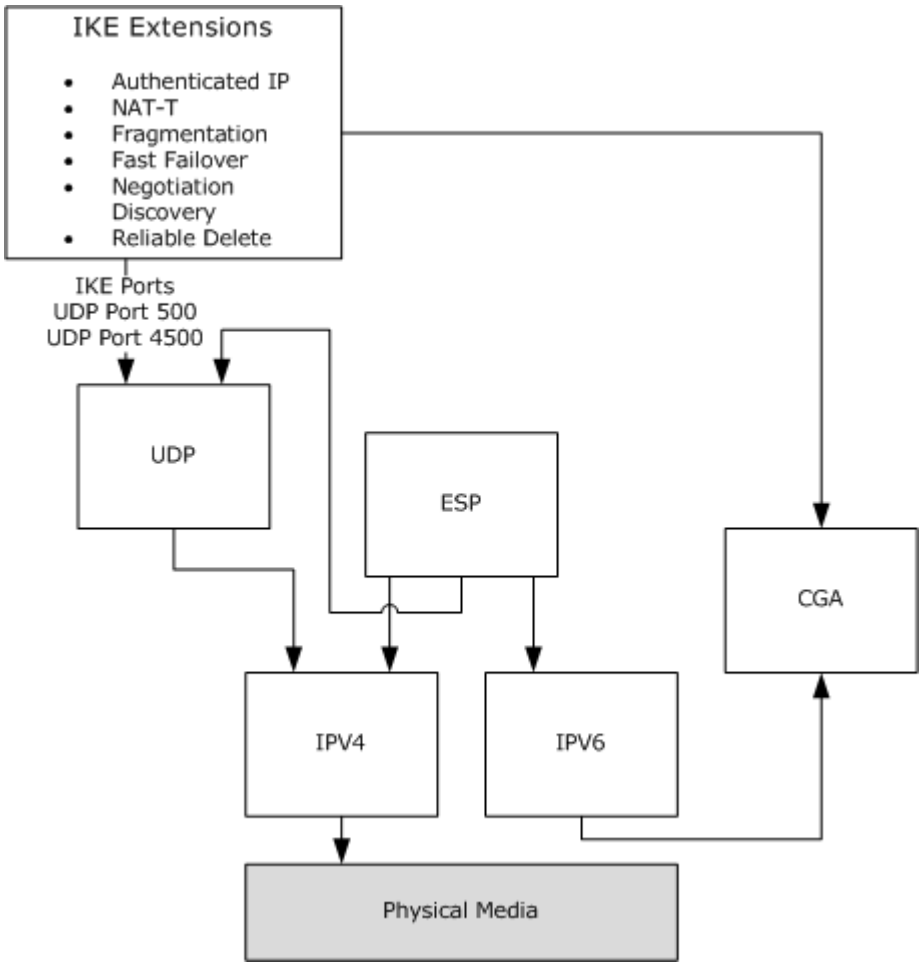


Figure 16: Scenario extensions

All but one of the extensions in the scenario consist of changes in the Internet Key Exchange (IKE) Protocol Extensions, as specified in [MS-IKEE].

The scenario alters the previously mentioned protocol stack in one respect only: it supports authentication by using a cryptographically generated address (CGA), as specified in [RFC3972]. This alteration does not introduce a new layer. Instead, it introduces an additional logical dependency between the IKE Protocol Extensions and the CGA functionality of the underlying Internet Protocol version 6 (IPv6) implementation.

Copyright © 2011 Microsoft Corporation.

3.1.3.1.2 Logical Dependencies

This scenario introduces a logical dependency between the Internet Key Exchange (IKE) Protocol Extensions and the IPv6 CGA functionality.

The IKE Protocol Extensions are GSS-aware. Therefore, their use of authentication services results in the logical dependencies as specified in section 3.1.

3.2 Domain Services Interaction

Domain Services Interaction encompasses both the ability of domain members to share the authentication and authorization services of the domain controller, and also the ability to administer and manage that authorization. Similar to the previous section on Base Authentication and Authorization, this topic is divided into conceptual areas, as shown in the following diagram.

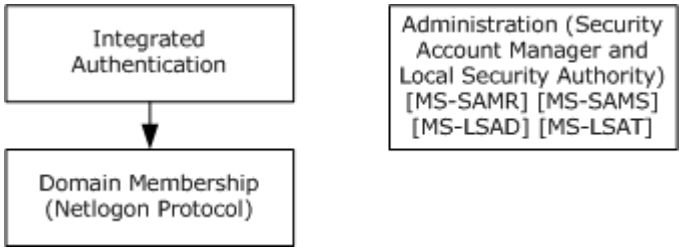


Figure 17: Basic authentication

The join-a-domain scenario is provided in Appendix A of this document.

3.2.1 Integrated Authentication

As shown in the Base Authentication and Authorization section, Integrated Authentication offers authentication services to various application protocols. At the receiving end of authentication, it is frequently useful to use a domain controller for authentication and authorization. Authentication Protocol Domain Support specifies how various authentication protocols connect to the domain controller and make use of domain resources.

3.2.1.1 Protocol Stack

Various aspects of domain-based authentication are displayed in the following diagram.

Copyright © 2011 Microsoft Corporation.

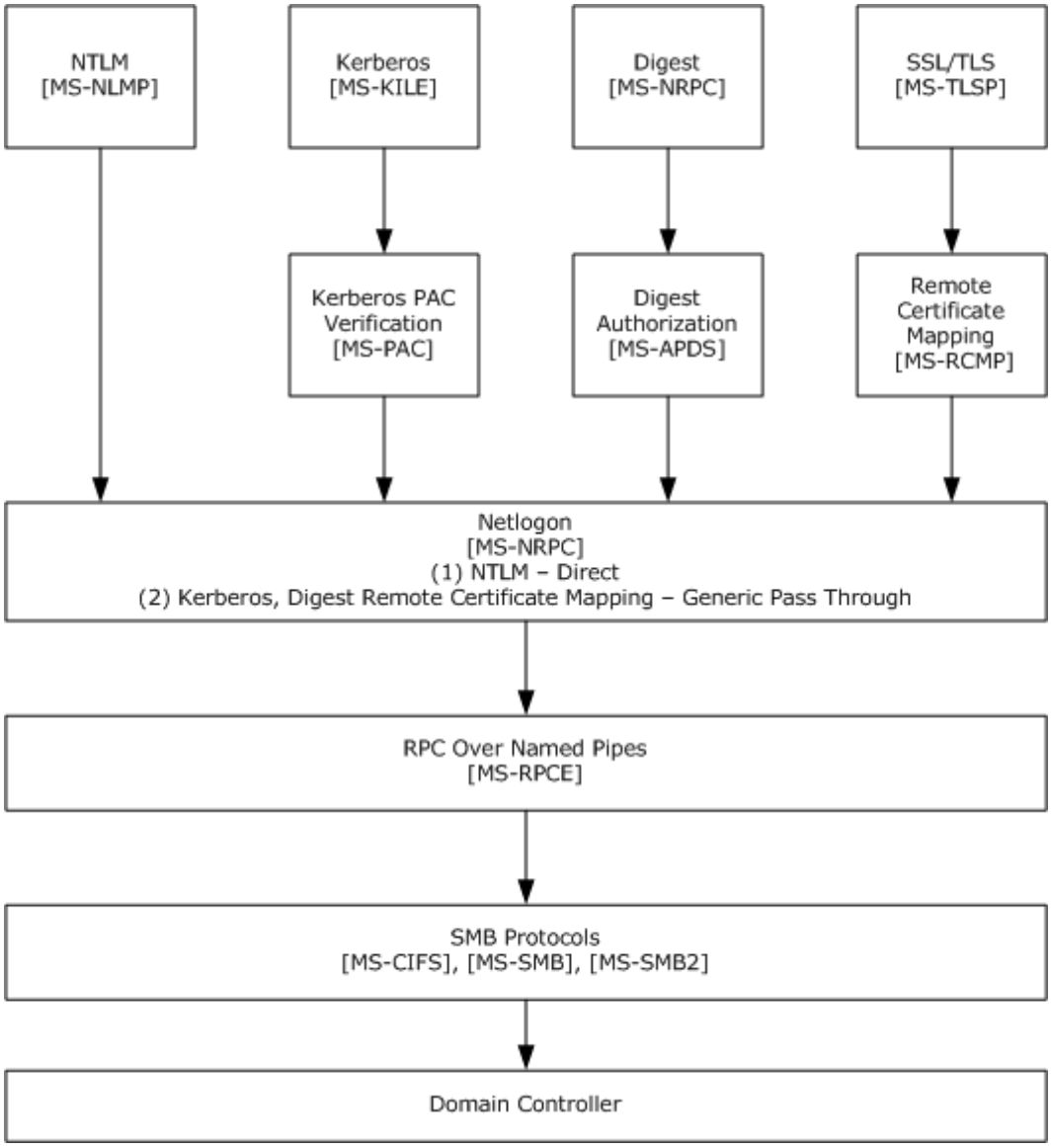


Figure 18: Domain-based authentication

In this diagram, the authentication protocols in the top row use the Netlogon Remote Protocol, either directly—in the case of the NTLM Authentication Protocol, or indirectly—through the generic pass-through facility of Netlogon. The latter allows the Remote Certificate Mapping Protocol, Digest Protocol Extensions, and Privilege Attribute Certificate (PAC) Data Structure all to occur at the domain controller.

3.2.2 Administration

Administration allows a user to manipulate user and group accounts, as specified in [MS-SAMR], and to translate names to identifiers and back again by using the Local Security Authority (Translation Methods) Remote Protocol.

Copyright © 2011 Microsoft Corporation.

3.3 Multifactor Authentication and Certificate Services

A group of protocols facilitate the use of Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol for two-factor authentication. A public key infrastructure (PKI) is one that allows people, computers, and services to be authenticated by using X.509 certificates. When use of such certificates is combined with the need to know a password or personal identification number (PIN) to access the private key that is related to the certificate, two-factor authentication is the result. The two factors are: something known (the password or PIN) and something possessed (the private key).

The protocols employed in this scenario are the following:

- The Remote Certificate Mapping Protocol, which is used by a server to contact a directory to determine authorization information, such as the group memberships of a client that authenticates itself by using an X.509 certificate.
- The Windows Client Certificate Enrollment Protocol (also known as the Certificate Services DCOM interfaces) allows a client to interact with a certificate authority (CA) for enrollment, issuance, and management of X.509 certificates in a PKI.
- The ICertPassage Remote Protocol, which exposes a Remote Procedure Call Protocol Extensions interface that allows a client to interact with a CA to request and receive X.509 certificates from a CA.

3.3.1 Protocol Stack - Remote Certificate Mapping Protocol

The Remote Certificate Mapping Protocol consists of a single request/response message pair: SSL_CERT_LOGON_REQ and SSL_CERT_LOGON_RESP. This request/response pair is transported by using the generic pass-through capability of the Netlogon Remote Protocol. Therefore, it depends on the Remote Procedure Call Protocol Extensions for its functionality. In turn, any protocol that uses X.509 certificates to authenticate a client to a server can result in the server using the Remote Certificate Mapping Protocol. The two common examples of such protocols are Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Internet Key Exchange (IKE).

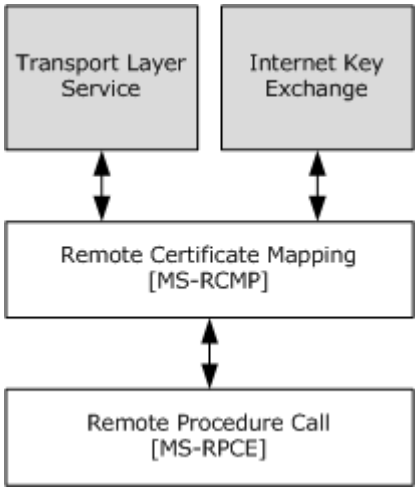


Figure 19: SSL/TLS and IKE

Copyright © 2011 Microsoft Corporation.

3.3.2 Protocol Stack - Client Certificate Enrollment Protocol

The Windows Client Certificate Enrollment Protocol uses the DCOM Remote Protocol, as specified in [MS-DCOM], as a transport. Therefore, it depends on the DCOM Remote Protocol to accomplish the task of allowing clients to manage its X.509 certificates through their entire life cycle. The protocol is used by clients for a variety of tasks that include querying a certificate authority (CA) for its capabilities, enrolling for certificates, requesting certificates on behalf of others, renewing certificates, backing up keys, placing keys in escrow, and archiving its keys.

Although many of the protocols that use PKI depend on these processes having been completed or accomplished in some manner, some protocols directly depend on this protocol or directly use it. Therefore, any of a variety of protocols can trigger activity that results in the client having to use this protocol but no protocols explicitly do so. Actually, local use of encryption based on asymmetric keys, such as Encrypting File System (EFS), can also require WSCCS-based enrollment and procurement of certificates.

Examples of protocols that use X.509 certificates include TLS/SSL, IPsec/IKE, and EFS.

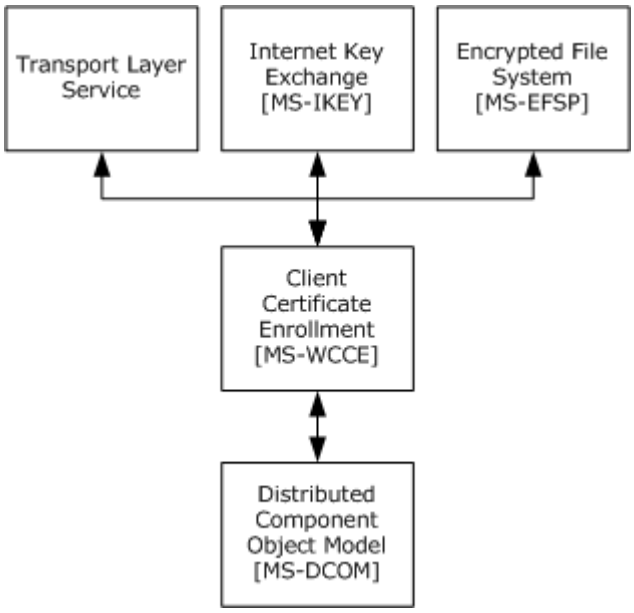


Figure 20: TLS/SSL, IPsec/IKE, and EFS

3.3.3 Protocol Stack - ICertPassage Enrollment Protocol

The ICertPassage Enrollment Protocol exposes a Remote Procedure Call Protocol Extensions interface that allows a client to interact with a certificate authority (CA) to request and receive X.509 certificates from a CA.

The ICertPassage Enrollment Protocol is a subset of the Windows Client Certificate Enrollment Protocol. The ICertPassage Enrollment Protocol provides only certificate enrollment functionality. The Windows Client Certificate Enrollment Protocol provides richer functionality, including reading and writing CA data and configuration information, in addition to the certificate enrollment functionality (as specified in [MS-WCCE]). The ICertPassage interface defines one method: CertServerRequest (as specified in [MS-ICPR]).

Copyright © 2011 Microsoft Corporation.

Note The ICertPassage Enrollment Protocol was first introduced in Microsoft Windows NT® 4.0 operating system as the first certificate enrollment protocol for a Microsoft Windows®-based platform. The Windows Client Certificate Enrollment Protocol was introduced in Microsoft Windows® 2000 Server operating system as the preferred certificate enrollment protocol and as a replacement for the ICertPassage Enrollment Protocol.

The ICertPassage Enrollment Protocol depends on the Remote Procedure Call Protocol Extensions, as specified in [MS-RPCE]. No other Windows protocol depends on ICertPassage.

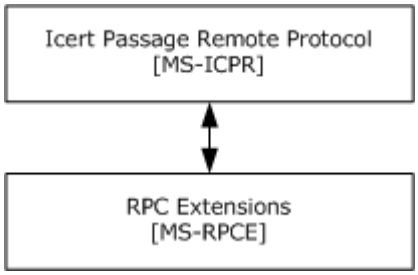


Figure 21: ICertPassage

3.3.4 Protocol Stack - Key Service Remote Interface Protocol

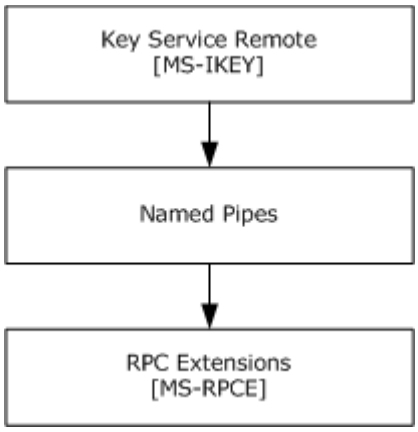
The Key Service Remote Interface Protocol delivers a certificate and its related private key to a trusted server. It does so via a Personal Information Exchange (PFX) file (for more information, see [PKCS12]) to the server. The PFX file contains the certificate and its associated private key.

The protocol depends on the use of mutual authentication capability in the Remote Procedure Call Protocol Extensions. The server registers the interface for the RPC at a well-known named pipe path so that an RPC client caller can bind to the interface and call its methods.

This protocol uses the following well-known endpoint. This endpoint is a pipe name for RPC over SMB: \PIPE\keysvc.

Because an RPC session must be authenticated, signed, and encrypted, there is a dependency on RPC_C_AUTHN_GSS_NEGOTIATE as the RPC security support provider (SSP), as specified in [MS-SPNG].

No Microsoft Windows® protocols depend on the use of this protocol. This protocol has been removed from Windows Vista® operating system.



Copyright © 2011 Microsoft Corporation.

Figure 22: Key service remote interface

3.3.5 Logical Dependencies

The protocols, taken together, form a set of services that meet the needs of PKI-using applications. Their logical interdependencies are represented in the following diagram.

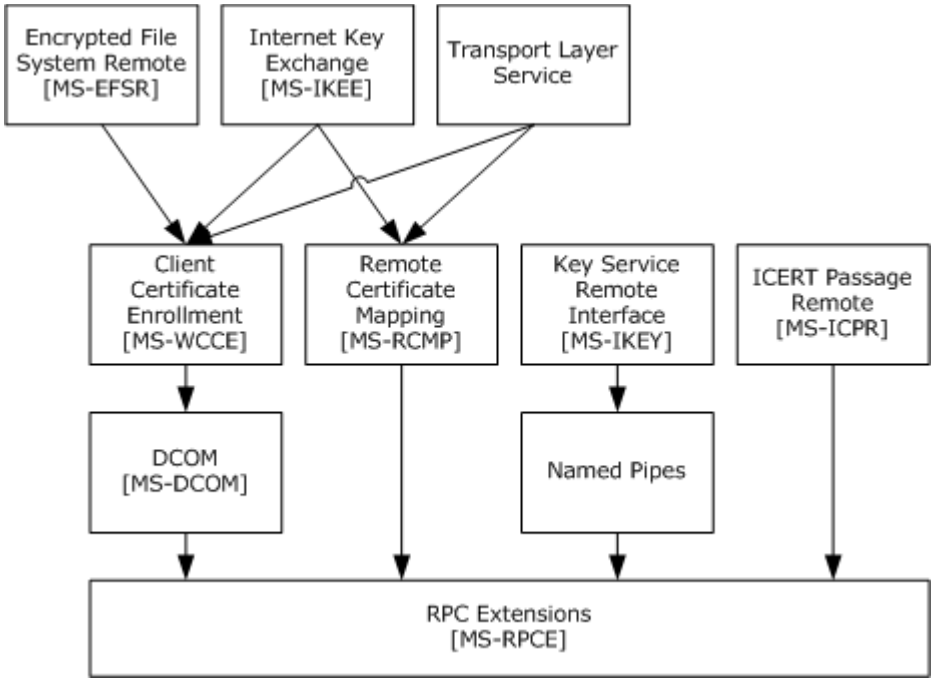


Figure 23: Protocol interdependencies

Note The protocols at the top of the diagram are simply PKI/X.509 certificate-using applications and only indirectly employ the protocols discussed here. Ultimately, they all depend on RPC and its ability to communicate securely across a network.

3.4 Group Policy

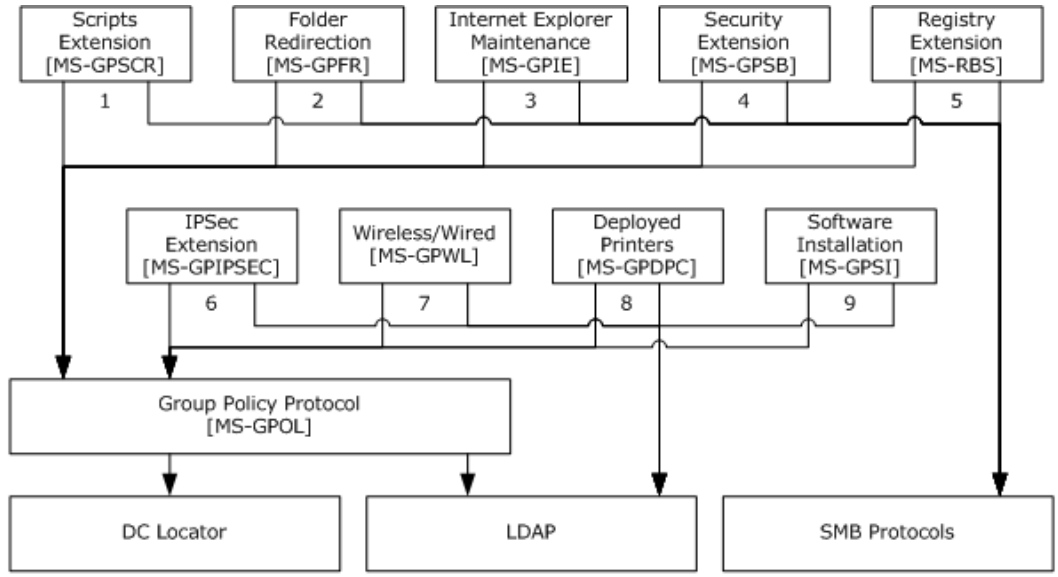
This section describes the relationship among protocols and protocol extensions that are used for Group Policy. The main purpose of this scenario is to configure computers in a domain with parameters that relate to various other components.

3.4.1 Protocol Stack

A Group Policy client uses both the Lightweight Directory Access Protocol (LDAP), as specified in [LDAP], and Server Message Block (SMB) as specified in [MS-SMB], to connect to a domain controller and retrieve metadata about the classes of available configuration data. The domain controller need not be explicitly aware that it is being used for Group Policy because it is merely a typical LDAP and SMB server. The client simply makes LDAP queries and SMB file copies.

The following figure depicts the protocol stack that is used by Group Policy.

Copyright © 2011 Microsoft Corporation.



#	CSE GUID	Tool Extension GUID: Computer Policy Settings	Tool Extension GUID: User Policy Settings
1	{42B5FAAE-6536-11D2-AE5A-0000F87571E3}	{40B66660-4972-11D1-A7CA-0000F87571E3}	{40B66650-4972-11D1-A7CA-0000F87571E3}
2	{A2E30F80-D7DE-11D2-BBDE-00C04F86AE3B} {4CFB60C1-FAA6-47f1-89AA-0B18730C9FD3}	n/a	{FC715823-C5FB-11D1-9EEF-00A0C90347FF}
3	{25537BA6-77A8-11D2-9B6C-0000F8080861}	n/a	{C40D66A0-E90C-46C6-AA3B-473E38C72BF2}
4	{827D319E-6EAC-11D2-A4EA-00C04F79F83A}	{803E14A0-B4FB-11D0-A0D0-00A0C90F574B}	n/a
5	{35378EAC-683F-11D2-A89A-00C04FBBCFA2}	{0F6B957D-509E-11D1-A7CC-0000F87571E3}	{0F6B957E-509E-11D1-A7CC-0000F87571E3}
6	{DEA8AFA1-CC85-11D0-9CE2-0080C7221EBD}	{DEA8AFA0-CC85-11D0-9CE2-0080C7221EBD}	n/a
7 (wireless)	{0ACDD40C-75AC-47AB-BAA0-BF6DE7E7FE63}	{2DA6AA7F-8C88-4194-A558-0D36E7FD3E64}	n/a
7 (wired)	{B587E2B1-4D59-4E7E-AED9-22B9DF11D053}	{06993B16-A5C7-47EB-B61C-B1CB7EE600AC}	n/a
8	{8A28E2C5-8D06-49A4-A08C-632DAA493E17}	{180F39F3-CF17-4C68-8410-94B71452A22D}	{180F39F3-CF17-4C68-8410-94B71452A22D}
9	{C6DC5466-785A-11D2-84D0-00C04FB169F7}	{7E45546F-6D52-4D10-B702-9C2E67232E62}	{1BC972D6-555C-4FF7-BE2C-C584021A0A6A}

Figure 24: Group Policy protocol stack

By itself, Group Policy is incapable of communicating policy settings directly and can merely communicate the metadata about what classes of settings are available. To retrieve actual configuration data, a client must be extended by one or more protocol extensions that are capable of communicating the policy settings of a specific class. These extensions are above Group Policy in the protocol stack, functioning as higher-layer entities (in the same way that TCP and UDP are above IP in a standard protocol stack).

Group Policy itself has no dependency on any such extensions (just like IP does not depend on TCP or UDP). Any number of extensions can be added without requiring changes to this protocol. Instead, extensions register themselves with Group Policy, just as server applications register themselves with TCP.

Copyright © 2011 Microsoft Corporation.

The extension mechanism is made to be easily extendible by any vendor. Microsoft provides a set of core extensions. Each extension is independent of the others and can use any transport and any server. Microsoft extensions, like Group Policy itself, use either LDAP or SMB to retrieve the configuration data that is stored on the domain controller.

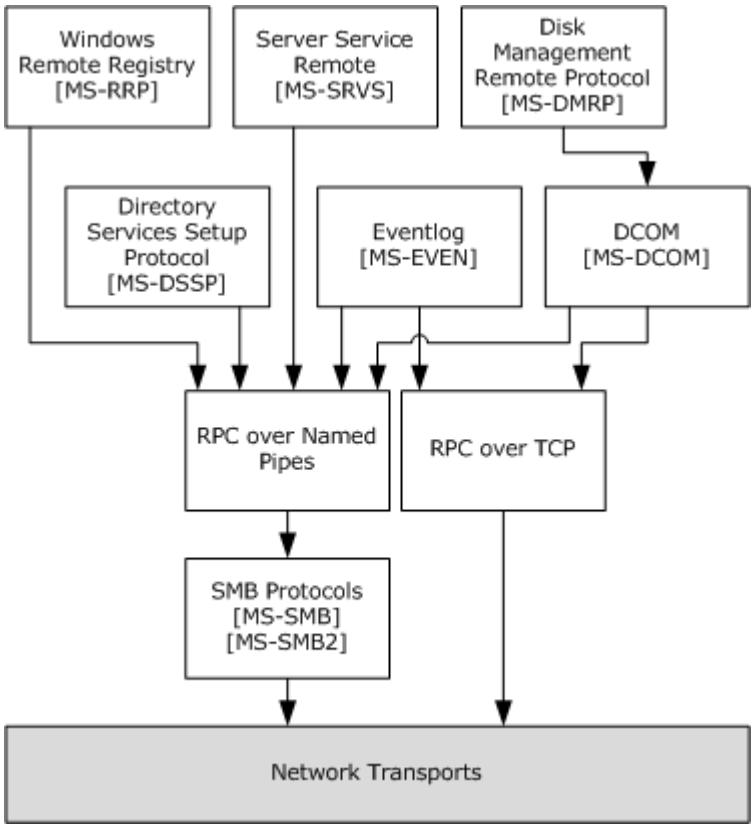
The same protocol stack also exists for a Group Policy administration tool, which can be used to author the configuration data and upload it for retrieval by clients. An extension authors and stores the configuration data of a specific class, and Group Policy authors and stores the metadata about the classes of the available data.

In addition to relying on LDAP or SMB, Group Policy also depends on the DC Locator protocol (as specified in [MS-ADTS] section 7.3) to locate the domain controller.

Each class of data is identified by one or more GUIDs, which Group Policy uses in the metadata to identify the classes of available data. There are generally separate GUIDs for the administrative tool, for policy data that applies to a computer, and for policy data that applies to a user. However, separate GUIDs are not required and not all classes of data apply to both users and computers. The set of GUIDs that are used by Microsoft are shown in the previous table (see the above figure), where the number in the first column indicates the dependency.

3.5 Systems and System Health Management

Systems and System Health Management is the ability to control settings and to collect data for a set of clients and servers. The protocols in this scenario are protocols that enable the client to either query another system or to manipulate that other system for administrative purposes. The overall scenario comprises several protocols, as shown in the following diagram.



Copyright © 2011 Microsoft Corporation.

Figure 25: Protocol scenarios

As can be seen in the preceding diagram, the Systems and System Health Management scenario is comparatively high-level; it runs over RPC or DCOM.

Several of the protocols in this scenario are covered elsewhere, for example:

- The Server Service Remote Protocol (SRVSVC) and the Disk Management Remote Protocol are covered in section 2.1.
- The Workstation Service Remote Protocol (WKSSVC) has its own scenario in the Network Connection Management scenario.
- The Windows Remote Registry Protocol is covered in section 3.8.
- Event logging also has its own scenario, as specified in section 3.9.

The Directory Services Setup Remote Protocol (DSSETUP) is used to query a remote computer about its domain-related status. The Directory Services Setup RPC interface is only accessible over an RPC *named pipes* binding.

3.5.1 Messenger Services

Messenger Services has two parts to it. The name management RPC works over a named pipe. The actual message delivery can happen over connectionless RPC, SMB, or mailslots, as shown in the following diagram.

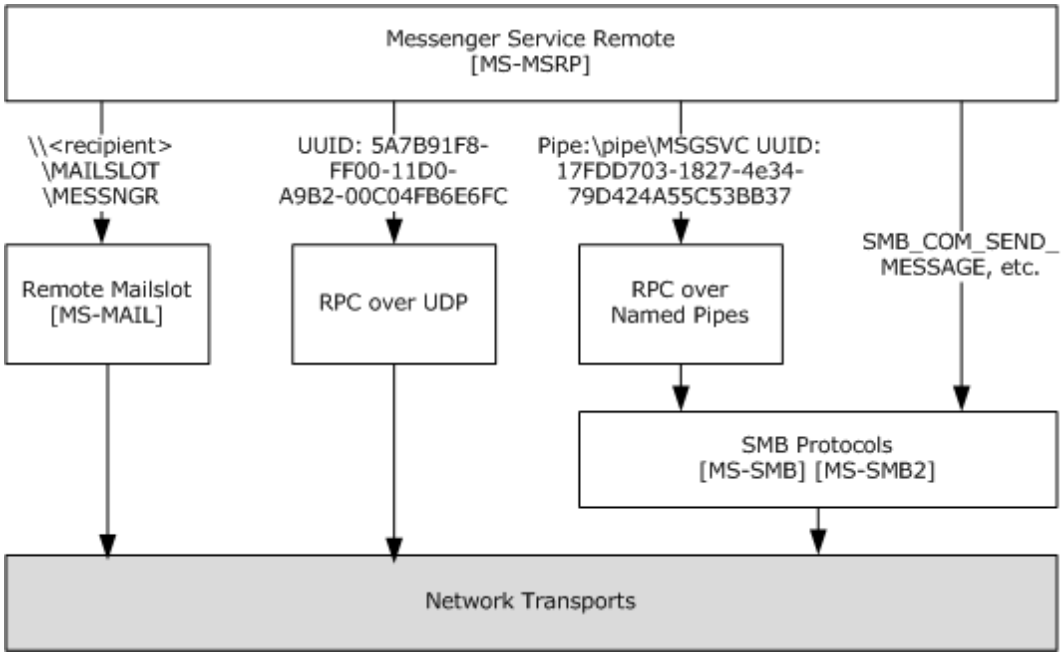
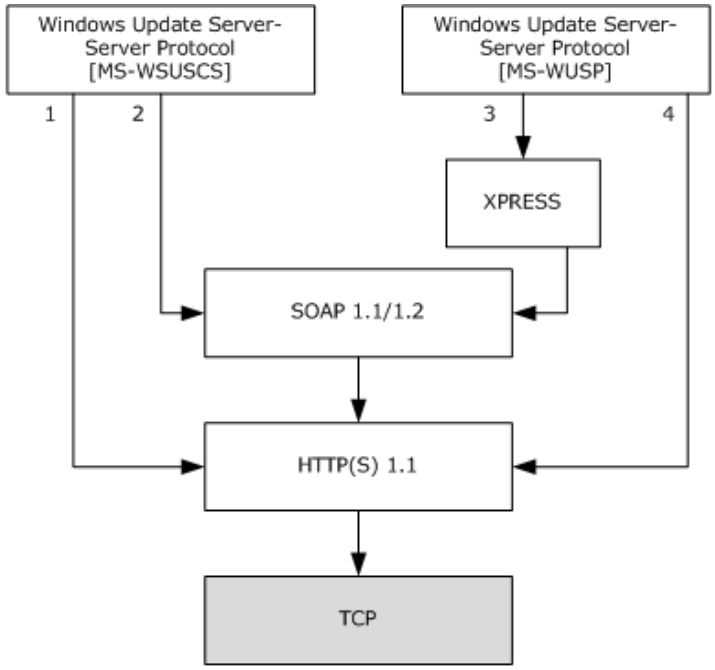


Figure 26: Messenger services protocol stack

Copyright © 2011 Microsoft Corporation.

3.5.2 Windows Update Services



1 – Content: http://server:<server port>/Content/<folder name>/<content file name>
2 – Server Sync: http[s]://<server>:<server port>/ServerSyncWebService.asmx
Authorization: http[s]://<server>:<server port>/dssauthWebService.asmx
3 – SimpleAuth Web Service: http[s]://serverUrl:[commonPort]/SimpleAuthWebService]/SimpleAuth.asmx
Client Web Service: http[s]://serverUrl:[commonPort]/ClientWebService/Client.asmx
Reporting Web Service: http[s]://serverUrl:[commonPort]/Reporting WebService/Reporting WebService.asmx
4 – Self-Update Tree: http://serverUrl:[commonPort]/SelfUpdate
Content Directory: http://serverUrl:[contentPort]/Content
Verion 0.9 Legacy: http://serverUrl:[commonPort]/ ..

Figure 27: Windows Update services

Windows Update Services allows administrators to manage the downloading of operating system upgrades for remote clients. One protocol is used to synchronize the patches and deployment data between the servers that provide the update service. A second protocol is for communication between clients and the servers that deploy update patches.

The Windows Update Services: Server-Server Protocol is implemented as two SOAP-based Web services that run exclusively over HTTP 1.1 as the transport. The actual patches are downloaded via HTTP from a content directory on that server.

The Windows Update Services: Client-Server Protocol consists of three SOAP-based Web services that also run exclusively over HTPP 1.1 as the transport. The metadata store and the actual patches are made available on two virtual directories for HTPP download. When the client calls the Client Web Service or SimpleAuth Web Service with "xpress," in the "Accept-Encoding" request-header, the server returns the result compressed in that format. The "xpress" encoding is as specified in [MS-DRSR].

Copyright © 2011 Microsoft Corporation.

3.6 Directory and Global Catalog Replication

Directory replication is as specified in the Active Directory Technical Specification and the Directory Replication Service (DRS) Remote Protocol. Active Directory servers also support a variation for replication based on the Simple Mail Transfer Protocol (SMTP) mail, as specified in [RFC2821]; this is available only for global catalog replication.

The DRS Remote Protocol is used to replicate information from a naming context on one instance of the directory to the same naming context on another instance; or from a subset of a naming context to a global catalog.

3.6.1 Protocol Stack

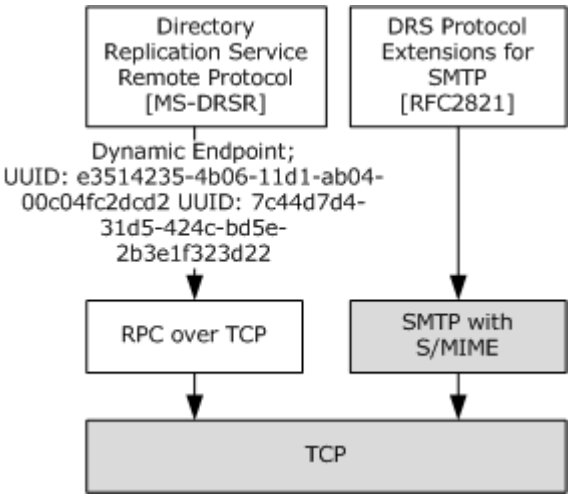


Figure 28: Protocol stack

In a domain, the Directory Replication Service (DRS) Remote Protocol is used. This protocol has a straightforward RPC interface that runs over TCP. The DRS Remote Protocol is used to transmit sensitive information, and therefore, it relies on the security services offered by the Kerberos protocol, as specified in section 3.1.

Between a domain and a global catalog server, a "store-and-forward" method of replication can be accomplished via SMTP. This is comparatively rare in actual deployment; typically, the DRS Remote Protocol is used for global catalogs as well.

3.6.2 Logical Dependencies

The most important logical dependency for these protocols is based on the data model of Active Directory, as specified in [MS-ADTS].

Copyright © 2011 Microsoft Corporation.

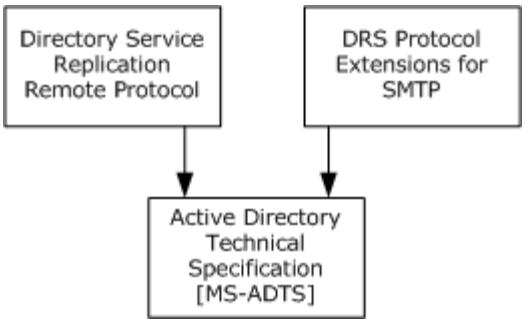


Figure 29: Directory replication protocols

The directory replication protocols are responsible for maintaining consistency among the data of different instantiations of Active Directory naming contexts; therefore, the directory replication protocols have a strong correlation with the overall data model of Active Directory.

3.7 Kerberos Group Membership

The Kerberos Group Membership scenario is not a protocol; it instead includes the Privilege Attribute Certificate (PAC) Data Structure that is used in Kerberos and other protocols to carry information about the group memberships of a particular user, potentially along with additional information. Other information can include credential information when used together with Public Key Cryptography for Initial Authentication in Kerberos (PKINIT), as specified in [RFC4556], authentication for the Kerberos protocol, and logon policy information. Full details of what can be carried in a PAC are as specified in [MS-PAC].

3.7.1 Logical Dependencies

Because Kerberos Group Membership is a structure, not a protocol, there is no protocol stack. However, the PAC Data Structure does have some logical dependencies, as shown in the following diagram.

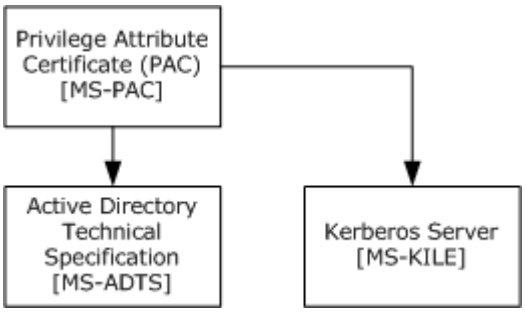


Figure 30: PAC Data Structure logical dependencies

Here, the PAC Data Structure depends on an account database, such as Active Directory (as specified in [MS-ADTS]), that contains the groups and other information that are used to populate the PAC Data Structure, as well as the policy to filter any identities that should not be propagated (as specified in [MS-PAC]). Similarly, the PAC Data Structure must be updated based on the surrounding Kerberos protocol, during Authentication Service (AS) ticket creation or ticket-granting service (TGS) ticket creation. This is as specified in [MS-KILE] and [MS-PAC].

Copyright © 2011 Microsoft Corporation.

3.8 Windows Remote Registry Protocol

The Windows Remote Registry Protocol, as specified in [MS-RRP], is used to manipulate the Microsoft Windows® registry on a remote computer. The preferred binding for the Windows Remote Registry Protocol is a named pipe; however, this interface is also available with ncacn_spx, ncacn_ip_tcp, ncacn_nb_nb, ncacn_nb_tcp, and ncacn_nb_ipx protocol sequence bindings.

3.8.1 Protocol Stack

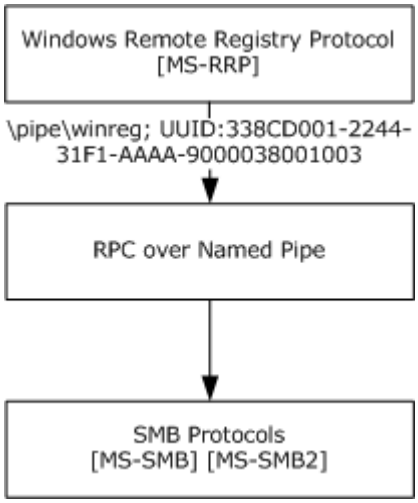


Figure 31: Protocol stack

The logical dependencies created by this arrangement are only those of the underlying SMB protocols that are used for the named pipes. In this case, the authentication services of SMB, as specified in section 2.1.2, are used by the Workstation Service Remote Protocol to authorize requests.

3.9 Eventlog Remote Protocol

This section describes the relationship among protocols and protocol extensions that are used for the Eventlog Remote Protocol scenario. The main purpose of this scenario is to read and write events to a log on a remote computer, and to remotely manage the set of logs.

3.9.1 Protocol Stack

This scenario has two versions of this protocol. The Eventlog Remote Protocol Version 1.0, as specified in [MS-EVEN], is the original protocol. The Eventlog Remote Protocol Version 6.0, as specified in [MS-EVEN6], obsoletes the original protocol and is now the preferred protocol.

Although both versions use RPC, as specified in [MS-RPCE], they use different RPC bindings, which results in a slightly different protocol stack, as shown in the following diagram.

Copyright © 2011 Microsoft Corporation.

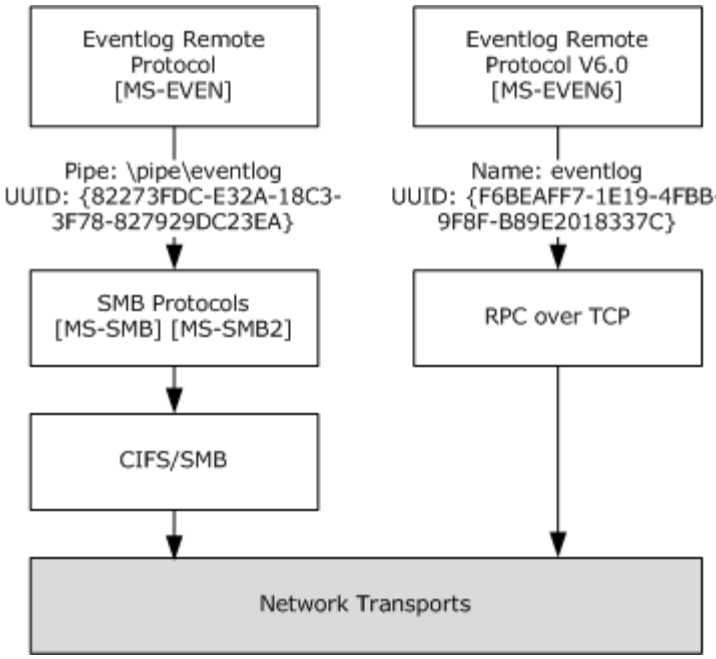


Figure 32: Eventlog Remote Protocol

The original protocol [MS-EVEN] exposes an RPC interface by using a named pipes binding. Named pipes are an abstraction that is offered by SMB. In contrast, the Eventlog Remote Protocol Version 6.0 exposes an RPC interface by using a dynamic endpoint that operates directly over TCP/IP, where dynamic endpoints are an abstraction that is offered by RPC itself.

The RPC layer performs the necessary encoding and decoding of the commands in both interfaces.

3.9.2 Logical Dependencies

Both versions of the Eventlog Remote Protocol, Eventlog Remote Protocol Version 1.0 and Eventlog Remote Protocol Version 6.0, use the authentication services of RPC, as specified in section 2.1. The Eventlog Remote Protocol Version 1.0 uses the SPNEGO Protocol Extensions or can use NTLM directly. The Eventlog Remote Protocol Version 6.0 can use the SPNEGO Protocol Extensions, or it can use Kerberos or NTLM directly.

3.10 Network Time Services

The Network Time Services scenario includes two protocols for synchronizing time and an RPC-based management protocol for managing the services that implement the time synchronization protocols.

3.10.1 Protocol Stack

In combination, the protocols in the Network Time Services scenario are layered as shown in the following diagram.

Copyright © 2011 Microsoft Corporation.

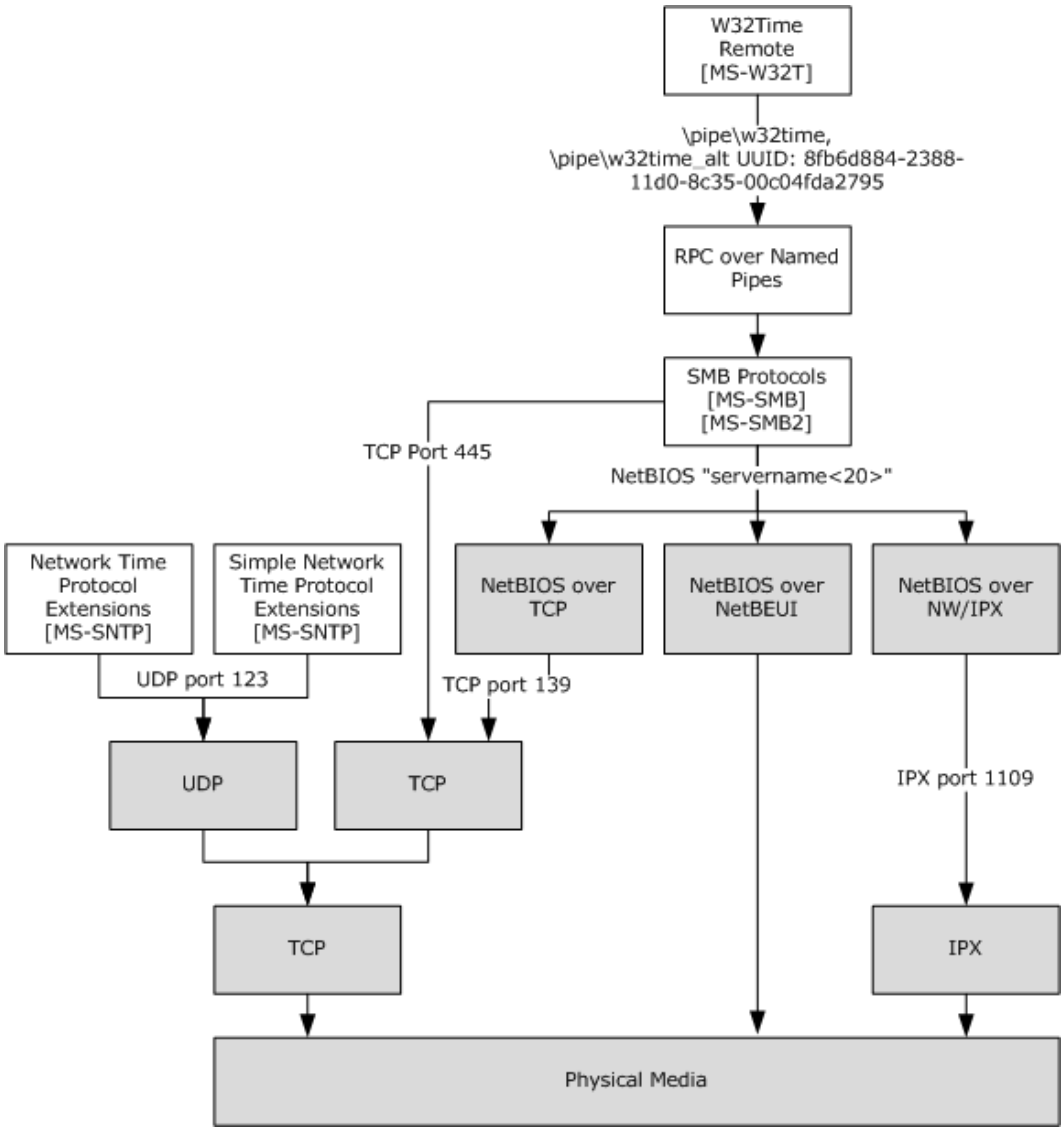


Figure 33: Layered protocols

The Network Time Protocol (NTP) Authentication Extensions apply to both the Network Time Protocol (NTP), as specified in [RFC1305], and the Simple Network Time Protocol (SNTP), as specified in [RFC1769]. Both the NTP Authentication Extensions and SNTP use UDP as their underlying transport.

The W32Time Remote Protocol consists of an RPC interface that provides methods for monitoring and controlling time services that implement the NTP Authentication Extensions. The RPC interface is accessible by using RPC over named pipes.

3.10.2 Logical Dependencies

The Network Time Services time synchronization protocol, as specified in [MS-SNTP], is not logically dependent on any protocols other than its underlying transports. However, a number of protocols in other scenarios depend on the services that they provide.

Copyright © 2011 Microsoft Corporation.

The Network Time Services management protocol, as specified in [MS-W32T], is an RPC interface that is accessible by using RPC over named pipes, which uses SMB as its underlying transport. As a result, the management protocol relies on the authentication services provided by SMB, as specified in section 2.1.2.

3.11 Network Connection Management

The Network Connection Management scenario is composed solely of the protocol that is used to instruct a node to create or otherwise manage an SMB connection to a server. This protocol, the Workstation Service Remote Protocol, is an RPC protocol that uses RPC over named pipes as its sole transport.

3.11.1 Protocol Stack

The Workstation Service Remote Protocol is a simple RPC interface that uses RPC over named pipes as its transport. The protocol stack for this protocol is a straightforward stack, as shown in the following diagram.

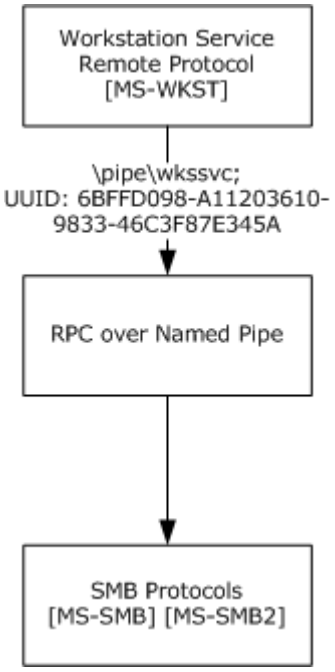


Figure 34: Workstation Service Remote Protocol stack

The logical dependencies that are created by this arrangement are only those of the underlying SMB protocols (the SMB Protocol and the SMB Version 2.0 Protocol) that are used for the named pipes. In this case, the authentication services of SMB, as specified in section 2.1.2, are used by the Workstation Service Remote Protocol to authorize requests.

3.12 Remote Procedure Calls

The Remote Procedure Calls scenario is composed of the following three protocols. The first two protocols consist of a set of extensions to their corresponding DCE/RPC specifications, and the third protocol specifies an encoding for error information.

Copyright © 2011 Microsoft Corporation.

- Remote Procedure Call Protocol Extensions: A set of extensions to the DCE 1.1 Remote Procedure Call Specification, as specified in [C706].
- Remote Procedure Call Location Services Extensions: A set of extensions and restrictions to the DCE Remote Procedure Call Location Services specification, as specified in [C705] and [C706].
- ExtendedError Remote Data Structure: An encoding for exchanging rich structured error information in a distributed system.

3.12.1 Protocol Stack

The protocols in this scenario are interrelated, and they overlap in how they use underlying transports.

The Remote Procedure Call Protocol Extensions are layered as shown in the following diagram.

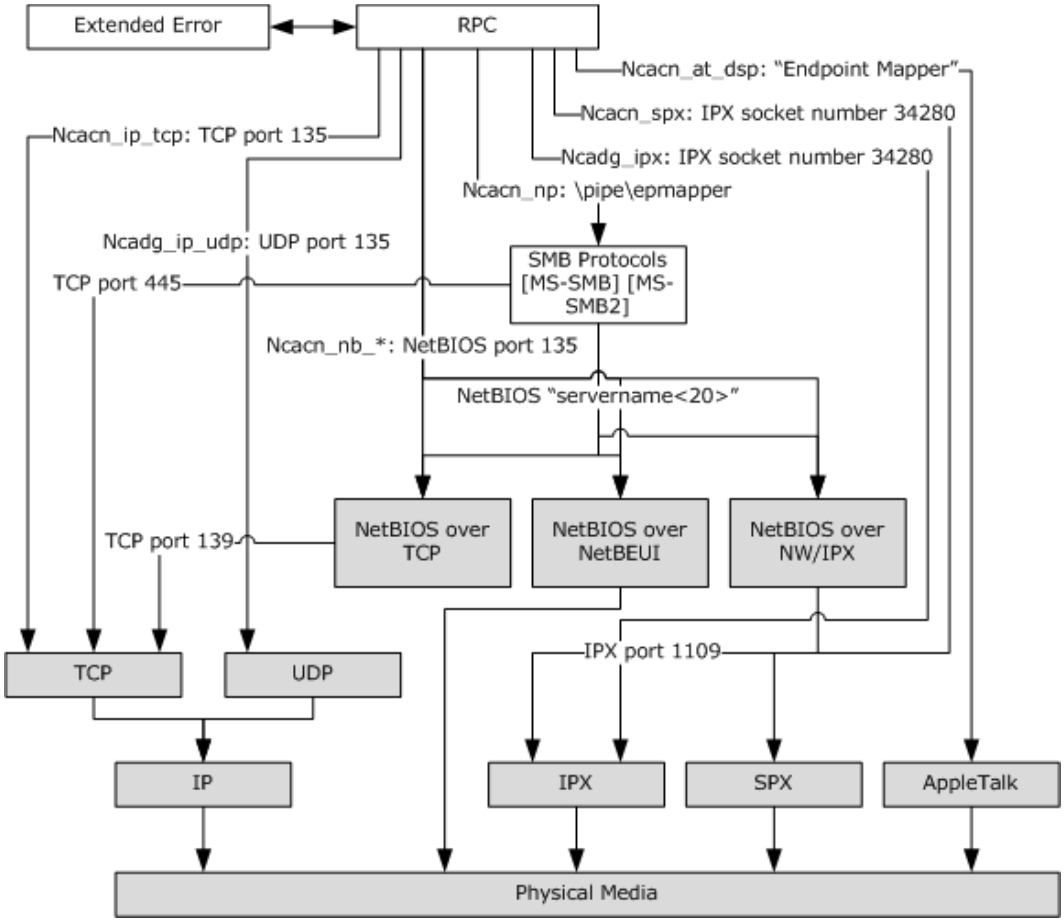


Figure 35: RPC Protocol Extensions layering

Remote Procedure Call Protocol Extensions define nine underlying protocol sequences that are layered over seven underlying transports. Of those seven underlying transports, two transports, SMB and NetBIOS, are layered over their own underlying transports. These two transports together are further layered over five underlying transports: TCP, IPX, NetBIOS over TCP, NetBIOS over

Copyright © 2011 Microsoft Corporation.

NETBEUI, and NetBIOS over NW/IPX. Those same five underlying transports are also the underlying transports over which Remote Procedure Call Protocol Extensions directly run.

The ExtendedError Remote Data Structure does not appear as a layer in the diagram because it consists only of an encoding that is employed by the Remote Procedure Call Protocol Extensions.

The Remote Procedure Call Location Services Protocol Extensions extend the preceding diagram with additional elements. The following diagram does not include details that are already provided in the preceding diagram (for example, it does not show the underlying transports that are used by SMB).

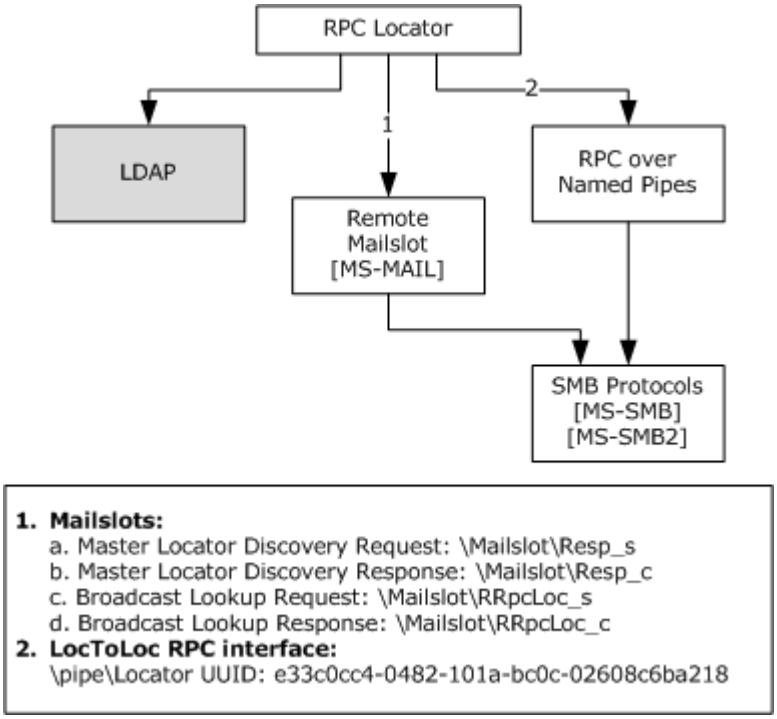


Figure 36: RPC Location Services Extensions

As shown in the preceding diagram, Remote Procedure Call Location Services Protocol Extensions employ three underlying transports:

- LDAP for access to RPC name service information in Active Directory, as specified in [MS-ADTS].
- The Remote Mailslot Protocol for master locator discovery and broadcast lookups.
- An RPC interface accessible through RPC over named pipes for invoking name service operations.

3.12.2 Logical Dependencies

Numerous other protocols depend on the facilities that are provided by the protocols in the Remote Procedure Calls scenario. This section identifies the protocols on which the protocols in this scenario are themselves dependent.

The Remote Procedure Call Protocol Extensions are not logically dependent on any protocols other than first, their underlying transports and second, the ExtendedError Remote Data Structure, which the Remote Procedure Call Protocol Extensions may use for exchanging error information.

Copyright © 2011 Microsoft Corporation.

The Remote Procedure Call Location Services Protocol Extensions have a number of dependencies in addition to their underlying transports, as show in the following diagram.

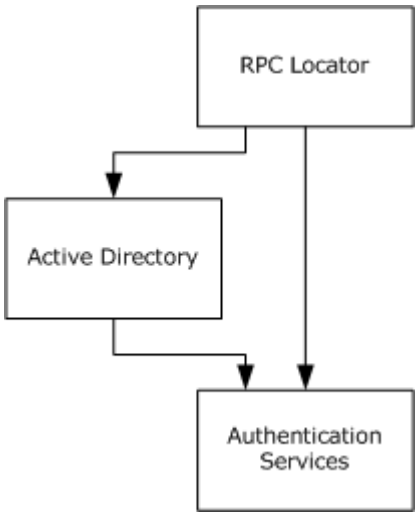


Figure 37: RPC dependencies

As previously mentioned, Remote Procedure Call Location Services Protocol Extensions rely on LDAP for access to the name service information in the Active Directory Technical Specification store.

Also, both Remote Procedure Call Location Services Protocol Extensions and Active Directory in turn rely on authentication services that are implemented through either LDAP or SMB. In the case of SMB, this reliance results in the dependencies as specified in section 2.1.2

3.13 Network Access Protection

The Network Access Protection scenario is a set of steps that are taken by a set of cooperating services that run on clients and servers to ensure policy compliance of computers in an enterprise network. This is typically done as the client is attempting to access the network.

In the Network Access Protection scenario, several protocols are used by various components within the client and the network access and authentication infrastructure.

The Health Certificate Enrollment Protocol and the Vendor-Specific RADIUS Attributes for Network Access Protection (NAP) Data Structure are central to Network Access Protection. The Vendor-Specific RADIUS Attributes for NAP Data Structure are extensions conformant with RADIUS extensions, as specified in [RFC2869], instead of an entire protocol. The protocol itself is RADIUS, as specified in [RFC2865]. Network Access Protection is a Statement of Health for Network Access Protection (NAP) Protocol.

The Health Certificate Enrollment Protocol is used by the client to prove its health to the network infrastructure and to get a certificate of good health. This certificate is an X.509 public key certificate, as specified in [RFC3280], that can then be used in any authentication protocol that relies on certificates and requires that the certificate also prove that the client is in "good health" (for example, in a state that conforms with network access policies of the enterprise).

RADIUS extensions are Microsoft's vendor-specific extension to the RADIUS messages that allow the protocol to become health-aware. RADIUS is an industry-standard protocol that is used by network access servers to authenticate clients that are accessing the network. The protocol itself runs

Copyright © 2011 Microsoft Corporation.

between the network access server (which, from the perspective of the RADIUS protocol, is the client) and a backend authentication server (the server in the RADIUS protocol).

3.13.1 HCEP Protocol Stack

The HCEP Protocol Stack is used to do the following:

- Convey information to a Health Registration Authority (HRA) regarding the state of the client.
- Request that a health certificate be issued.
- Get a response from the HRA that may include a certificate or remediation measures.

To accomplish this, the Health Certificate Enrollment Protocol uses HTTP, as specified in [RFC2616]. For the exchanges that contain confidential information, HTTP runs over Transport Layer Security (TLS), as specified in [RFC2818]. HTTP is the only protocol that the Health Certificate Enrollment Protocol directly uses. HTTP uses TCP as the transport protocol underneath it. The Health Certificate Enrollment Protocol, in turn, is invoked via an API by the Quarantine Agent (QA).

The HCEP Protocol Stack is run by a component called the health certificate enrollment agent (HCEA), which in turn, communicates over COM with applications that require health certificate enrollment.

On the client side, the protocol stack can be depicted as in the following diagram.

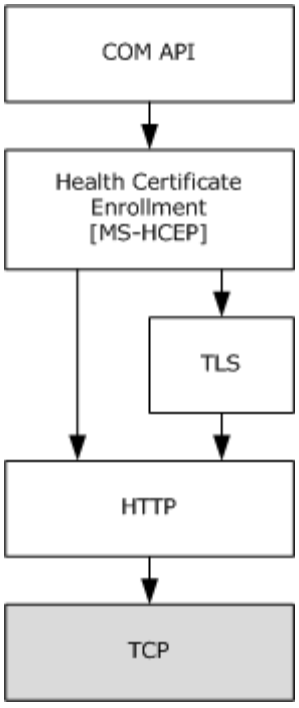


Figure 38: Client-side protocol stack

On the server side, the HCEP Protocol Stack is implemented in the Health Registration Authority (HRA), which communicates with the Internet Authentication Service (IAS)/**network policy server (NPS)** over RADIUS.

Copyright © 2011 Microsoft Corporation.

3.13.2 Logical Dependencies

There are other protocols that are used to implement a Network Access Protection system that do not appear in layers. These are parallel events that cause the Health Certificate Enrollment Protocol and the Vendor-Specific RADIUS Attributes for Network Access Protection (NAP) Data Structure with Microsoft's vendor-specific extensions to be used.

Any of the following events can cause the Health Certificate Enrollment Protocol to be used:

- The system health agent (SHA) determines that because of the expiry of a timer or a change in the policy, the health state needs to be recertified.
- The health state has changed.
- The previously obtained health certificate is close to expiry.
- A new IP address has been obtained by the system. This is notification of a system event.

When any of these events occur, a cooperating set of COM objects uses the Health Certificate Enrollment Protocol to refresh the health certificate. The health state is communicated to the Health Registration Authority (HRA), a response is received, and the certificate is deposited in the certificate store. The certificate is now available to the Internet Key Exchange (IKE), as specified in [RFC2409], which uses the certificate to prove the identity and good health to the system.

The following diagram outlines the protocols that are above and below the Health Certificate Enrollment Protocol.

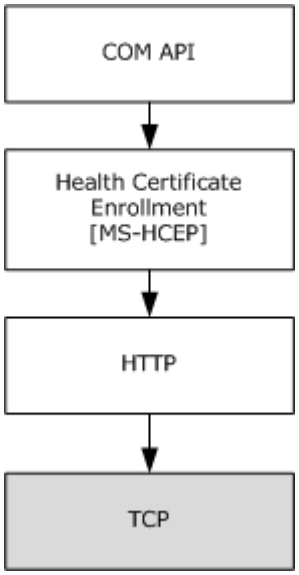


Figure 39: Logical dependencies of the Health Certificate Enrollment Protocol

On the server side of the Health Certificate Enrollment Protocol, communication is by means of the HRA. The HRA receives messages from the client over HTTP and it needs to confirm that the health of the client conforms to policy. It uses the RADIUS Protocol, which operates over UDP ports 1645, 1646, 181, and 1813, to reach an authentication and policy management server, and it authenticates the health claims as well as credentials of the HCEP client.

Copyright © 2011 Microsoft Corporation.

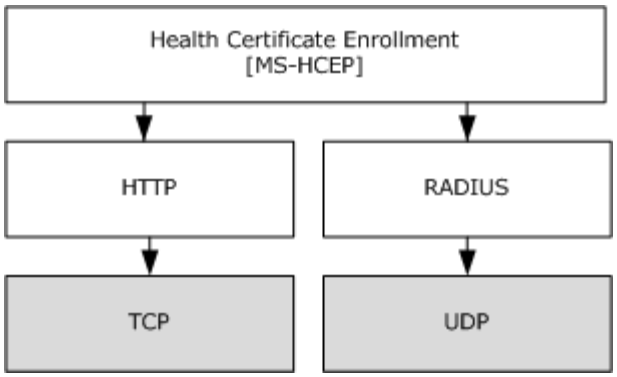


Figure 40: Health certificate enrollment

The Internet Authentication Service (ISA)/network policy server (NPS) is able to check the information that is communicated by the HRA to determine whether the user and the computer possess an authorized identity and are in a conformant state. On the basis of this conclusion, they communicate a RADIUS Access Accept or an Access Reject message to the HRA, which in turn, either issues a certificate or provides steps that the client needs to take to conform with policy.

3.13.3 RADIUS Extensions - Protocol Stack

The RADIUS Extensions are proprietary extensions to RADIUS messages. They are part of the RADIUS attributes in the Vendor-Specific RADIUS Attributes for Network Access Protection (NAP) Data Structure, and they run over UDP. The following diagram shows the protocol stack, and its dependent and using relationships.

The RADIUS client typically is an access point, a DHCP server, or a server running Routing and Remote Access. The RADIUS server is the authentication server that has the ability to connect to a directory or other authentication and policy management service.

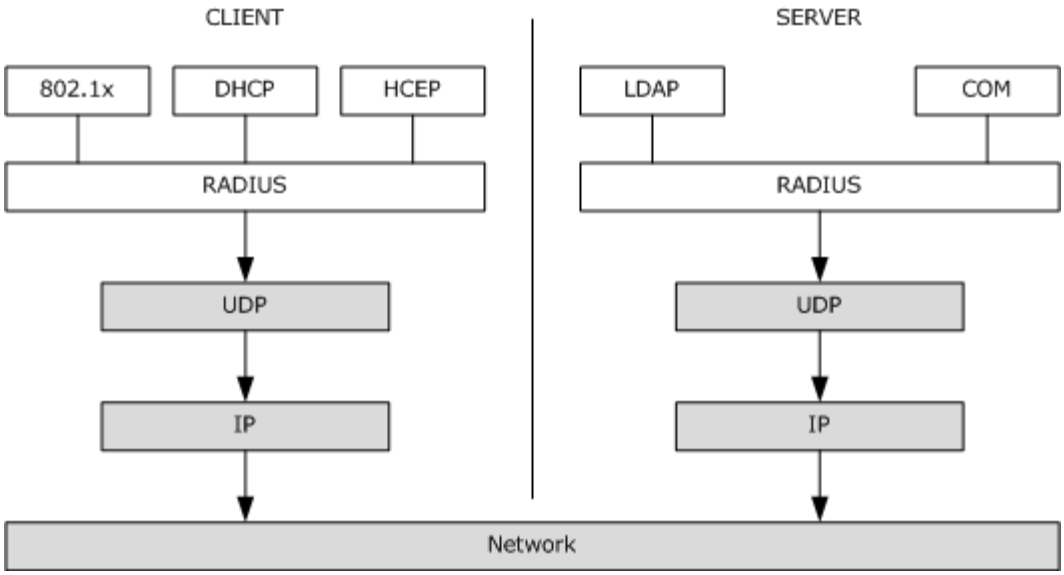


Figure 41: RADIUS protocol stacks

Copyright © 2011 Microsoft Corporation.

The RADIUS Extensions are simply elements in RADIUS messages that are vendor specific. These are called vendor-specific attributes (VSAs) and have been used here to add the ability to communicate health-related information in what was otherwise an authentication, authorization, and accounting (AAA) protocol.

Copyright © 2011 Microsoft Corporation.

4 Networking and Transports Task

The Networking and Transports Task contains no specific scenarios; it instead contains a variety of protocols that are general purpose or not easily categorized.

4.1 Protocols

A number of protocols that appear in the Networking Task have been previously specified, including:

- The Passport Server Side Include (SSI) Version 1.4 Protocol, as specified in Base Authentication and Authorization.
- The Eventlog Remote Protocol Version 1.0 and Eventlog Remote Protocol Version 6.0, as specified in Systems and System Health Management.
- The ExtendedError Remote Data Structure, as specified in Remote Procedure Calls.
- The ICertPassage Remote Protocol, as specified in Multifactor Authentication and Certificate Services.
- The Messenger Service Remote Protocol, as specified in Systems and System Health Management.
- The NTP Authentication Extensions and W32Time Remote Protocol, as specified in Network Time Services.
- RPC and secure RPC, as specified in Remote Procedure Calls.
- The Windows Remote Registry Protocol, as specified in Windows Remote Registry Protocol.
- The Workstation Service Remote Protocol, as specified in Network Connection Management.
- The Internet Protocol Security Protocol, as specified in Base Authentication and Authorization.

4.2 RPC over HTTP Protocol

The Remote Procedure Call over HTTP Protocol specifies the use of either HTTP (as specified in [RFC2616]) or HTTPS (as specified in [RFC2818]) as a transport for the Remote Procedure Call Protocol Extensions. In particular, the protocol specifies provisions for using HTTP request/response streams as virtual channels; for encoding rules for transporting RPC protocol data units (PDUs) with HTTP requests and responses; and for roles for participants in the protocol.

4.2.1 Protocol Stack

The following diagram illustrates the protocol stack that results from the combined roles and encoding rules for the Remote Procedure Call (RPC) over HTTP Protocol.

Copyright © 2011 Microsoft Corporation.

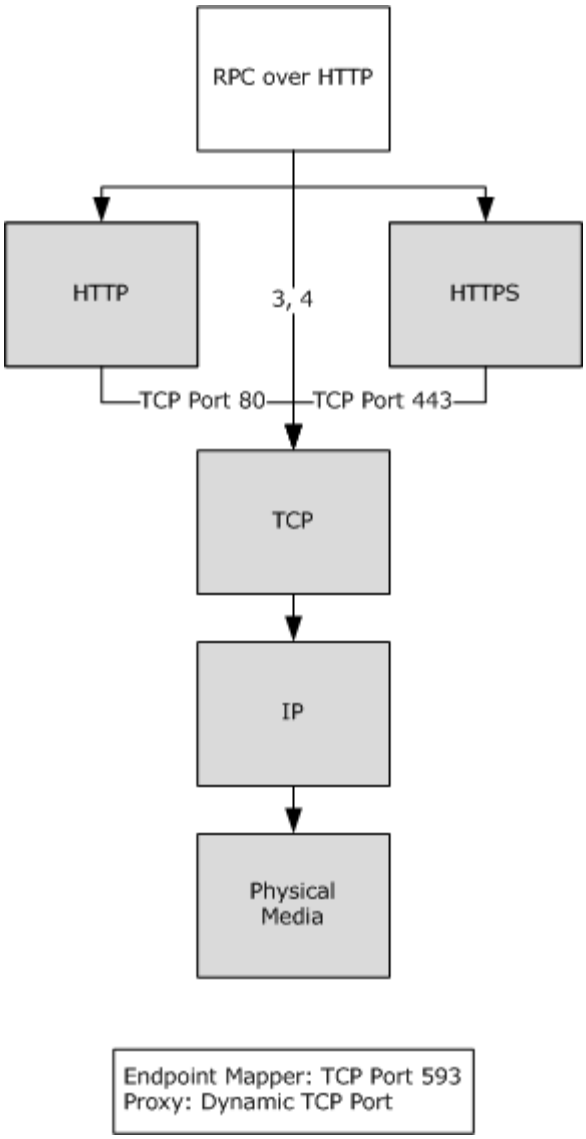


Figure 42: RPC over HTTP Protocol stack

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

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

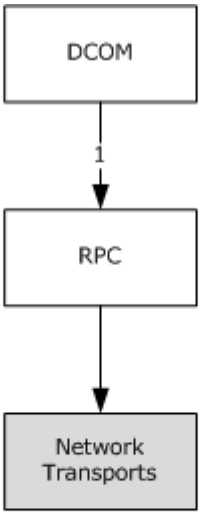
4.2.2 Logical Dependencies

The RPC over HTTP Protocol is not logically dependent on any protocols other than HTTP, HTTPS, and TCP, which are its underlying transports.

Copyright © 2011 Microsoft Corporation.

4.3 Distributed Component Object Model

The Distributed Component Object Model 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. This layering is illustrated in the following diagram.



1 – **RPC Interfaces** (all RPC protocol sequences):

- IActivation: 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
- IRemoteSCMActivator: 000001A0-0000-0000-C000-000000000046
- IObjectExplorer: 99fcfec4-5260-1010b-bbcb-00aa0021347a
- IRemUnknown: 00000131-0000-0000-C000-000000000046
- IRemUnknown2: 00000143-0000-0000-C000-000000000046

Figure 43: RPC protocol layering

The Distributed Component Object Model (DCOM) Remote Protocol is not logically dependent on any protocols other than the Remote Procedure Call Protocol Extensions. The dependencies for the latter are as specified in Remote Procedure Calls.

4.4 EAP MS-CHAPv2

This section describes the relationship among protocols and protocol extensions that are used for EAP MS-CHAPv2 authentication.

4.4.1 Protocol Stack

An EAP MS-CHAPv2 peer uses the Microsoft Challenge Handshake Authentication Protocol Version 2 (MS-CHAPv2), as specified in [RFC2759], run over the Extensible Authentication Protocol (EAP), as specified in [RFC2284] and [RFC3748], and a lower-layer protocol, such as PPP, as specified in [RFC1661], or [IEEE802.1X], to authenticate to an EAP server by using a network access server (also known as the EAP authenticator) as a pass-through. The EAP authenticator passes through EAP packets that are communicated between the EAP MS-CHAPv2 peer and server, using the RADIUS/EAP protocol, as specified in [RFC2759]; keying material is passed from the RADIUS/EAP server to the authenticator, using the Microsoft RADIUS VSA extensions, as specified in [RFC2548].

Copyright © 2011 Microsoft Corporation.

In turn, the RADIUS server uses the Netlogon Remote Protocol to communicate to a domain controller and validate the user credentials provided.

The relationship between the EAP peer, the EAP authenticator/network access server, and the EAP/RADIUS server is shown in the following figure.

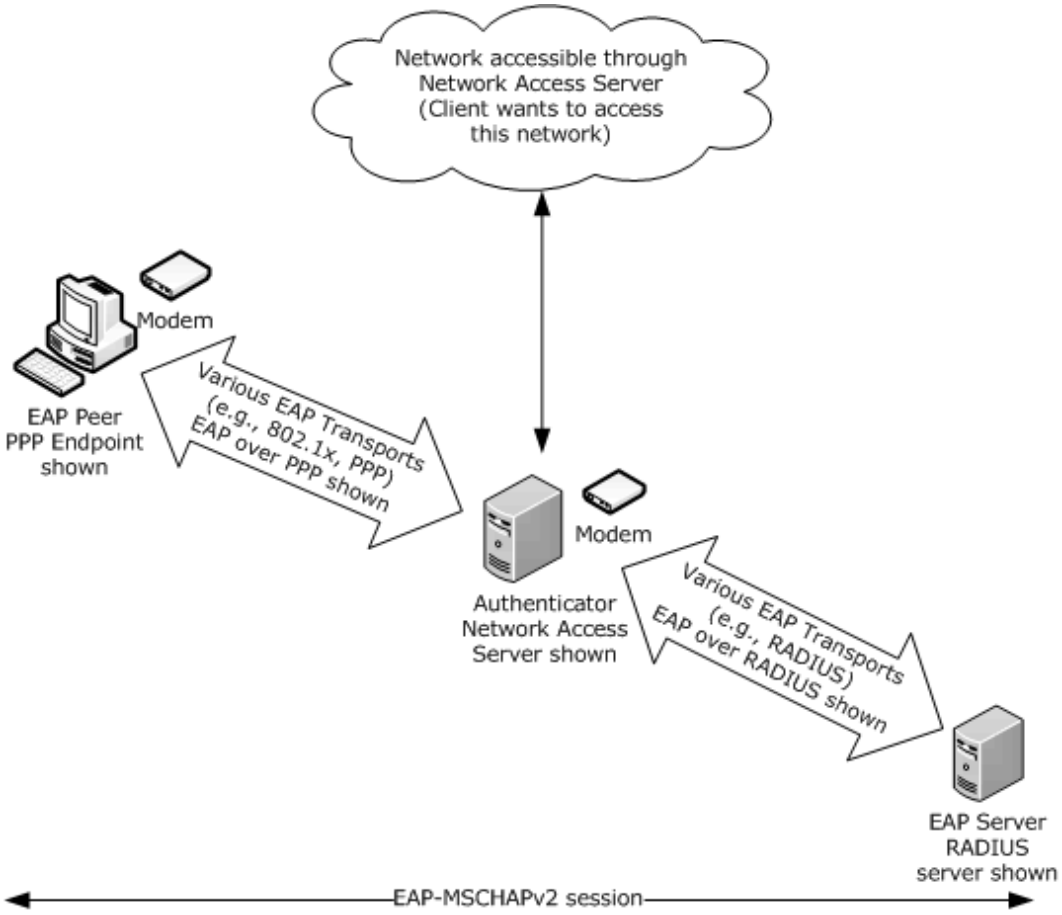


Figure 44: Relationship between the EAP peer, the network access server, and the EAP/RADIUS server

The following figure shows the protocol stack that is used by the EAP MS-CHAPv2 client.

Copyright © 2011 Microsoft Corporation.

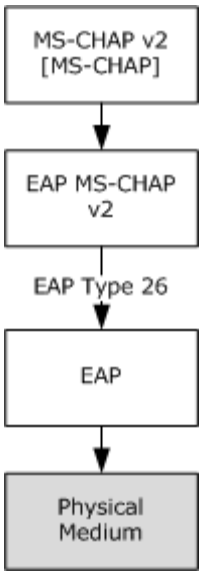


Figure 45: Client protocol stack

The following figure shows the protocol stack that is used by a RADIUS server.

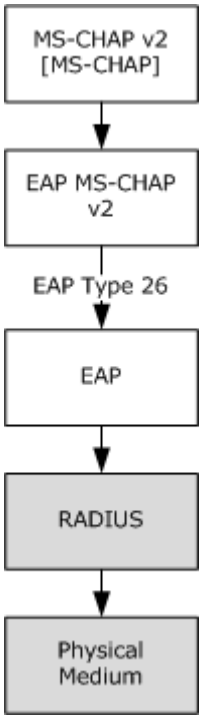


Figure 46: Server protocol stack

4.5 PEAPv0

This section describes the relationship among protocols and protocol extensions that are used for PEAPv0 authentication.

Copyright © 2011 Microsoft Corporation.

4.5.1 Protocol Stack

A PEAPv0 peer uses the Transport Layer Security (TLS) Protocol, as specified in [RFC2246], run over the Extensible Authentication Protocol (EAP), as specified in [RFC2284] and [RFC3748], and a lower-layer protocol, such as the Point-to-Point Protocol (PPP), as specified in [RFC1661] or [IEEE802.1X], to establish a server-authenticated TLS tunnel to an EAP server by using a network access server (NAS), also known as the EAP authenticator, as a pass-through. Within the TLS tunnel, an inner EAP method then runs to enable the peer to authenticate to the server. The EAP authenticator passes through EAP packets communicated between the PEAPv0 peer and server, using the RADIUS/EAP protocol, as specified in [RFC3579]; keying material is passed from the RADIUS/EAP server to the authenticator by using the Microsoft RADIUS VSA extensions, as specified in [RFC2548].

The relationship between the EAP peer, the EAP authenticator/NAS, and the EAP/RADIUS server is shown in the following figure.

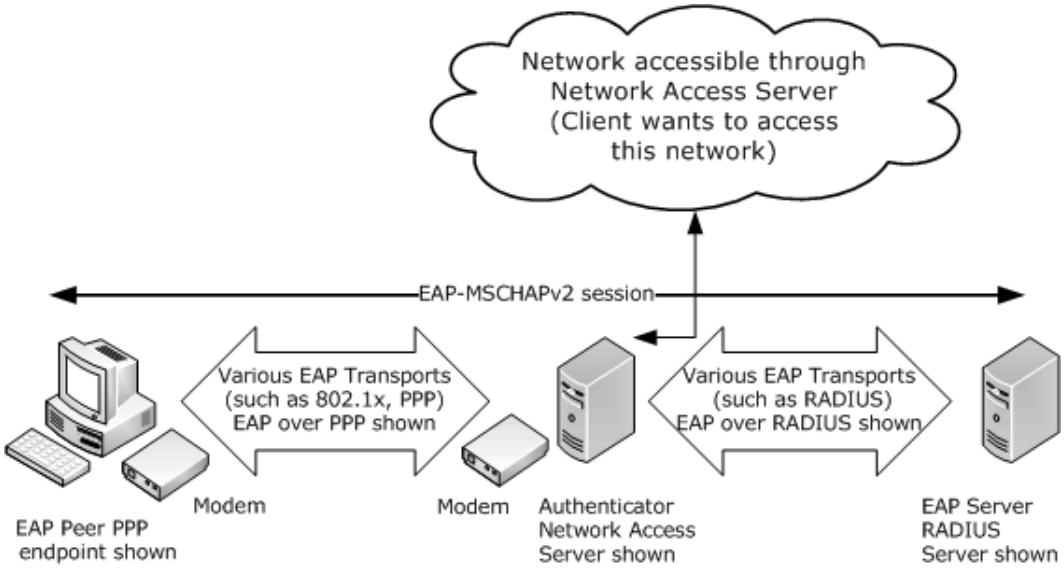


Figure 47: Relationship between the EAP peer, the network access server, and the EAP/RADIUS server

The following figure depicts the protocol stack that is used by the PEAPv0 client.

Copyright © 2011 Microsoft Corporation.

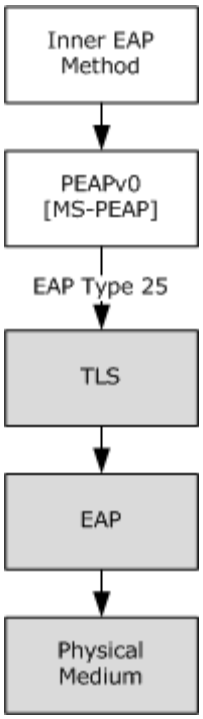


Figure 48: PEAPv0 peer protocol stack

The following figure depicts the protocol stack that is used by a RADIUS server.

Copyright © 2011 Microsoft Corporation.

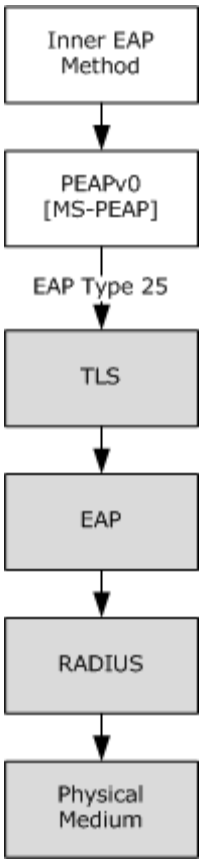


Figure 49: PEAPv0 server protocol stack

4.6 Internet Protocol Security Protocols Extensions

The Internet Protocol Security Protocols Extensions scenario is composed of the following elements:

- A set of extensions to the Internet Key Exchange (IKE) Protocol Extensions, as specified in [MS-IKEE].
- The Authenticated Internet Protocol, which peers may use instead of the IKE Protocol Extensions when negotiating IP security parameters.

4.6.1 Protocol Stack

The IPsec Protocols Extensions scenario retains the protocol stack, as specified in [RFC2409] and [RFC3947], for the IKE and IKE NAT-Traversal protocols, respectively.

Copyright © 2011 Microsoft Corporation.

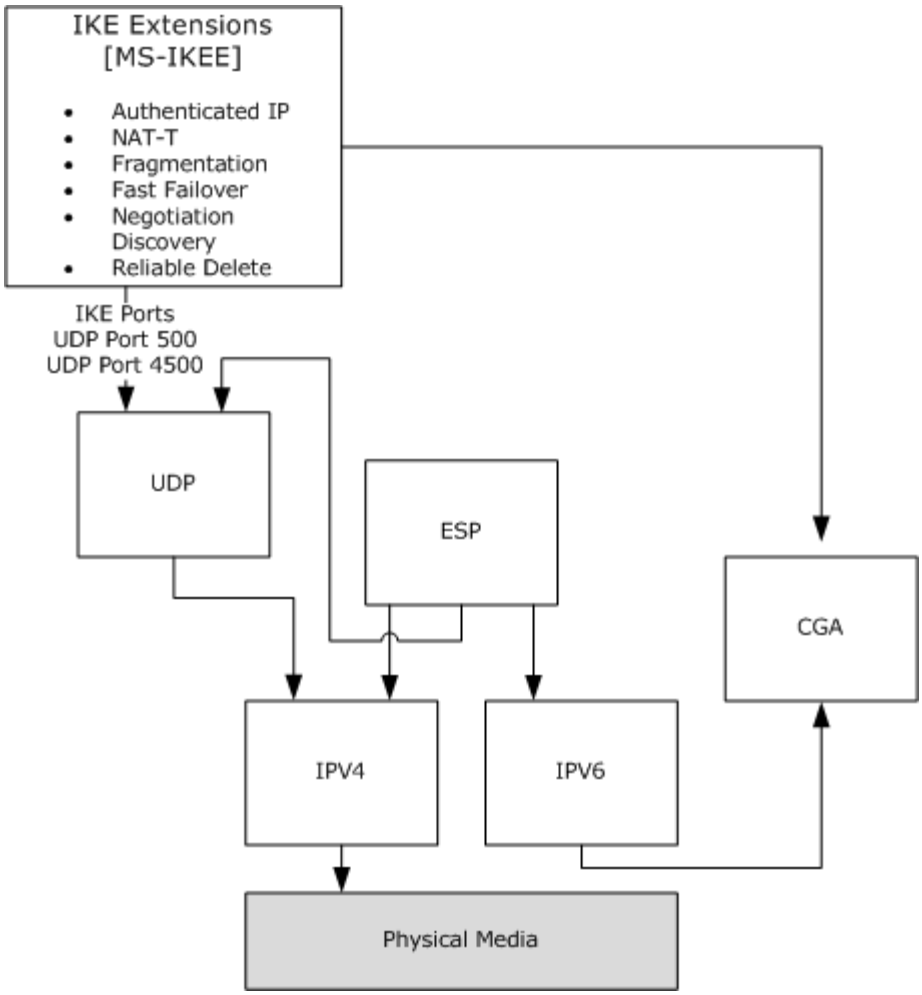


Figure 50: IKE Protocol Extensions

As shown in the preceding figure, all but one of the extensions in the scenario consist of changes in the IKE protocol itself.

The scenario alters the previously mentioned protocol stack in one respect only: it supports authentication by using a cryptographically generated address (CGA), as specified in [RFC3972]. This alteration does not introduce a new layer. Instead, it introduces an additional logical dependency between the IKE extensions and the CGA functionality of the underlying IP version 6 (IPv6) implementation.

4.6.2 Logical Dependencies

This scenario introduces a logical dependency between the Internet Key Exchange (IKE) Protocol Extensions and the IPv6 cryptographically generated address (CGA) functionality.

The IKE Protocol Extensions are a Generic Security Service (GSS)–aware protocol. As a result, use of these extensions for authentication services results in logical dependencies, as specified in section 3.1.

Copyright © 2011 Microsoft Corporation.

4.7 Windows Management Instrumentation

Windows Management Instrumentation, as specified in Windows Management Instrumentation Remote Protocol and the Windows Management Instrumentation Encoding Version 1.0 Protocol, describes the encoding of object types that are used in the protocol. The Windows Management Instrumentation Encoding Version 1.0 Protocol is used to access and manipulate remote management objects that are implemented according to the Common Information Model (CIM). For more information about CIM, see [DMTF-DSP004]).

4.7.1 Protocol Stack

The Windows Management Instrumentation Remote Protocol uses different default security levels in different versions of Microsoft Windows®, as specified in [MS-WMI]. For compatibility with all clients, the user can specify NTLM, the Kerberos protocol, or the SPNEGO Protocol Extensions.

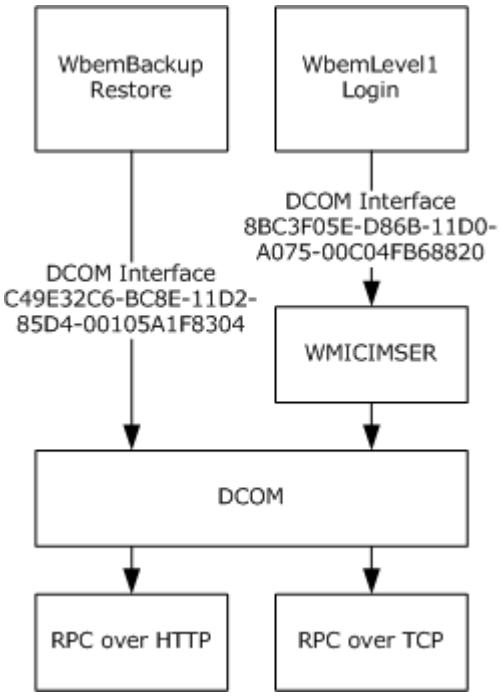


Figure 51: Protocol stack

Copyright © 2011 Microsoft Corporation.

5 WSPP Protocols Implementation Scenarios

This section provides 15 examples, or implementation scenarios, for several groups of the WSPP protocols that are covered earlier in this document.

5.1 Overview of the Implementation Scenarios Lab Configuration

The lab configuration for the implementation scenarios that follow used a private TCP/IP network with no background traffic. Network traffic was monitored and analyzed by using an Ethereal network analyzer.

A Windows Server® 2003 operating system computer was promoted to a domain controller in a new domain and configured as a DNS server role with full DNS services. The name of the domain is nttest.microsoft.com. The name of the domain controller is DC-1 and its fully qualified domain name is dc-1.test.com. A network capture for this setup is included in scenario 14.

A second Windows Server® 2003 R2 operating system computer was set up and promoted to a domain controller called DC-2.test.com. A network capture for this setup is included in scenario 15.

A third server was configured as a File and Print server role with network shares available and a printer shared to the network.

All protocols in these examples use TCP as the network transport. As with standard Internet Protocol communications, there were also communications of Address Resolution Protocol (ARP), User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP) data packets. Network captures of these packet types was omitted in these implementation scenarios because they are irrelevant to the purposes of the examples demonstrated here. However, the presence of these packets causes the number of packets captured in these examples to be only approximations.

The implementation scenarios do not fully document the packet forms for each individual protocol because these are documented in the relevant protocol specifications.

5.2 Scenario 1 - Add a Computer to a Domain

5.2.1 Scenario Overview

Total Packets: 1134

As specified in the Windows Security Overview, in the machine account of a domain, machine membership is separate from a user account. This separation of machine accounts allows machine-level policies to be applied in addition to a policy that is applied when a user logs on. The account separation provides the ability to authenticate that the machine is a member of the domain.

For future reference, this account is called MACHINE1. In the following case, the machine is initially in a work group. MACHINE1 was running Windows Vista® operating system Beta 2.

5.2.2 Procedure

1. The Domain Administrator logged on to MACHINE1, which had recently received a clean installation and was a member of the workgroup "Workgroup". The machine is named workgroup\ MACHINE1. The user account is a local account, MACHINE1 \USER1. At this point, the system has used the Common Internet File System (CIFS) Browser Protocol, DNS, and [LDAP].

2. USER1 clicked Start, clicked Control Panel, and then double-clicked System. USER1 then clicked the Computer Name tab and clicked the Change button (which was beside the text "To rename

Copyright © 2011 Microsoft Corporation.

this computer or join a domain, click Change"). In the dialog box, USER1 selected the domain option, entered the domain information (Test.com), and clicked OK.

3. USER1 was presented with a domain challenge dialog box where the relevant domain credentials had to be entered (test\Administrator and the Administrator's password).

4. At this point, most of the traffic between the client and server occurred. Because MACHINE1 had been entered into a new domain, the security identifier (SID) of MACHINE1 will need to change. Also, the domain administrators group will be added to the local administrators group. During that period the Server Message Block (SMB) Protocol, the NTLM Authentication Protocol, the Remote Procedure Call Protocol, and the Security Account Manager (SAM) Remote Protocol (Client-to-Server) were used. USER1 was then prompted to restart MACHINE1.

   Note When this was tested, approximately 300 packets were captured. This is where the machine's new SID was committed.

5. After the system was restarted, the user was presented with two options: log on as the previous local account or log on as the other user account. The user selected "other user" to enter the domain account information TEST\ADMINISTRATOR.

   Note When this was tested, a CIFS Browser Protocol announcement of the new host name occurred.

6. After the TEST\ADMINISTRATOR logged on, new settings flowed to the client by using the SMB Protocol, [LDAP], the Security Account Manager (SAM) Remote Protocol (Client-to-Server), the DRS Remote Protocol, the NTLM Authentication Protocol, the FRS Protocol, and the Netlogon Remote Protocol until the interactive session had ended.

7. If policy had been enabled, as is the default, the SMB traffic, while the ADMINISTRATOR logged on, would primarily be Group Policy: Core Protocol traffic to set the local policy on MACHINE1 and ADMINISTRATOR.

8. After the logoff, replication of the new settings occurred between the DCs by using [LDAP] and the DRS Remote Protocol.

5.3 Scenario 2 - Add a User to a Domain

5.3.1 Scenario Overview

Total packets: 1526

This scenario was one where the ADMINISTRATOR added a user to the domain as a member of the "normal users" group. Membership in that group account provided the user with rights to log on locally to any machine in the domain by using user-level access privileges.

5.3.2 Procedure

1. The TEST\ADMINISTRATOR logged on to an interactive session on the Microsoft Windows®-based FPS server.

2. The TEST\ ADMINISTRATOR opened the "Active Directory Users and Computers" option from Administrative Tools in the Start menu of the FPS Windows-based server. This action opened the Microsoft Management Console (MMC) and created traffic between FPS and DC1.

Copyright © 2011 Microsoft Corporation.

Note When tested, this step consisted of 720 packets, mostly of [LDAP] Generic Security Service Application Programming Interface (GSS-API)–encrypted information for the MMC to pre-cache for the Administrator to have a reasonable experience in MMC.

3. The ADMINISTRATOR opened the hierarchy in the navigation pane (the left pane) to the Active Directory Users and Computers/Domain name/Users container.

   Note Because this information was cached, it does not cause any significant traffic between the FPS server and DC1.

4. The ADMINISTRATOR right-clicked the mouse on the Users container and selected the option "New User." A dialog box appeared that contained the relevant entries for a new user account.

5. The ADMINISTRATOR filled in the relevant information and clicked Create. The first SMB packets appeared where the FPS server was connected to the \\DC1.test.com\IPC$ management share. The trailing $ indicated a hidden share. Netlogon traffic to enumerate domain trusts occurred. Then [LDAP], DRSUAPI, and Kerberos transactions added the user to the domain.

   Note When tested, the last three steps resulted in approximately 130 packets of information.

6. DC1 communicated the information to DC2 by using a combination of transactions over [LDAP] and directory replication.

   Note This step communicated less than 100 packets of information.

5.4 Scenario 3 - Users First Logon

5.4.1 Scenario Overview

Total packets: 286

When a user logs on to a machine in a domain, certain user information is automatically downloaded to the machine: user information such as Group Policy settings, application data, and a local cache of user documents. Also, the domain administrator typically retains the default option, which requires users to change their temporary passwords on first logon so that users can select their own password.

5.4.2 Procedure

1. The TEST\USER1 logged on to MACHINE1 by pressing CTRL+ALT+DEL, by selecting the Other User option, and by entering domain\user information and a temporary password that was provided by the administrator.

   Note Traffic flowed from MACHINE1 to the domain controller. A policy that USER1 must change the temporary password forced the next step.

2. The user received a dialog box that required the user to type the old user password, type the new user password, and then type the new password again to confirm the entry. The process of changing the password created a burst of Kerberos and NetLogon activity, which passed information to the SAM server on DC1.

   Note Many organizations also have security policies to make sure the user creates a secure password, such as requiring a minimum length for a password or requiring a combination of letters, numbers, and capitalization. Security policies such as these create more traffic between the client and the domain controller.

Copyright © 2011 Microsoft Corporation.

Note At this point in the logon process, about 120 of the 286 packets have been used.

3. Over the ensuing period of time, policy for USER1 and other options the administrator had set (such as pre-installing applications) flowed to MACHINE1.

   Note When tested, the minimal amount of options was set. This amounted to about 100 packets of information, mainly [LDAP] and SMB traffic.

4. Soon after, the user was ready to use the session.

5.5 Scenario 4 - Connect to a File Share Using GUI

5.5.1 Scenario Overview

Total packets: 212

In this scenario, the user right-clicked Network in the Start menu and selected the "Map a Network Drive" option. This action created a drive mapped as Drive Z on MACHINE1 for USER1's session.

5.5.2 Procedure

1. USER1 selected the Start menu and then selected Network. This action created SMB and [NTLM] traffic that ended in a set of DRSUAPI calls.

   Note This packet capture was approximately 110 packets.

2. USER1 opened the FRS server and selected the share "file" in the Map option for the file share. It is mapped to drive Z on MACHINE1.

3. This action created a connection to the FRS server. The typical [LDAP], TCP, and RPC packets—along with the first Workstation Service Remote Protocol packets—appeared on the wire.

   Note At about packet 190, the share was mapped.

5.6 Scenario 5 - Search for a File on the File Server

5.6.1 Scenario Overview

Total packets: 23

This scenario is the shortest; it shows a basic SMB transaction, a set of query information that looks at attributes, file attributes that flow across the wire (the latter to check the user ACLs), and a response to the DIR command.

5.6.2 Procedure

USER1 opens a command shell and enters "Dir z:\a.txt".

Note This packet capture was the smallest capture of all the scenarios.

5.7 Scenario 6 - Copy a File from FRS to the Local Machine

5.7.1 Scenario Overview

Total packets: 56

Copyright © 2011 Microsoft Corporation.

This was a simple scenario that was all SMB traffic. It started with a standard set of SMB queries to the remote server; it continued with a file lock request (at about 25 packets into the sniff), which was followed by a standard SMB copy command and a file close request.

5.7.2 Procedure

USER1 copies z:\a.txt to MACHINE1.

Note A total of 56 packets were captured.

5.8 Scenario 7 - Print a File on a Domain Shared Printer

5.8.1 Scenario Overview

Total packets: 297

USER1 printed a file to a printer that was shared to the network. In this case, an HP PhotoSmart 1315 printer was connected to the FPS server. A printer queue was shared out from that server. Because Windows Vista® operating system provides a printer driver for the HP PhotoSmart 1315, there was no need to provision a printer driver.

5.8.2 Procedure

1. USER1 clicked the Start menu and selected Network.

   Note This action created some SMB and [NTLM] traffic that ended in a set of DRSUAPI calls.

2. USER1 selected the HP PhotoSmart 1315 printer from the FRS server.

   Note This action created some spooler, SMB, and RPC traffic, before ending in an open request after about packet 100. Another set of spooler traffic then occurred and some SMB traffic, which ended in a SpoolerSS close request. The file that USER1 printed was small, 4 kilobytes (KBs), and this was less than 150 additional packets, mostly from SpoolerSS.

5.9 Scenario 8 - Delete a File from a File Share

5.9.1 Scenario Overview

Total packets: 696

USER1 navigated to the FRS server through the Network option in the Start menu. Because of the extra amounts to be rendered, this action was more expensive on the wire than scenario 5.

5.9.2 Procedure

1. User 1 connected to the file share.

   Note In the case of the writers sniff, this was about 250 packets of information.

2. When USER1 selected a.txt and deleted it by using the DELETE key on the keyboard, another typical SMB transaction took place that ended in a close request.

   Note The close request occurred at about 150 packets.

3. Another set of spooler traffic then occurred and some SMB traffic, which signaled to the machines that use the printer that the queue is flushed.

Copyright © 2011 Microsoft Corporation.

5.10 Scenario 9 - Log Off the USER1 Interactive Session

5.10.1 Scenario Overview

Total packets: 97

5.10.2 Procedure

In this scenario, USER1 logged off, and session and background replication took place. The traffic that took place is a combination of the Server Message Block (SMB) Protocol, the Security Account Manager (SAM) Remote Protocol (Client-to-Server), and the Kerberos protocol.

Note This sniff was just less than 100 packets; however, as with the user logon scenario, the size of this transaction depends on the options set by the administrator.

5.11 Scenario 10 - Log On as Domain Administrator

5.11.1 Scenario Overview

Total packets: 338

5.11.2 Procedure

1. Before the administrator logged on, any locally cached information was cleared from the ADMINISTRATOR Users previous logon cache.

2. After the cached information was cleared from MACHINE1, the sniffer produced data that was almost identical to the data in Scenario 3, without the change password.

3. The TEST\ADMINISTRATOR logged on to MACHINE1 by using the CTRL+ALT+DEL sequence, selecting the Other User option (because USER1 was the most recent user on that machine), and entering the domain information and password.

   Note Traffic flowed from MACHINE1 to the domain controller, ending with a set of Netlogon requests.

4. The ADMINISTRATOR's policy and other options then flowed to MACHINE1.

   Note In this test, a minimum amount of options was set and this transaction resulted in about 100 packets of information, mainly [LDAP] and SMB traffic.

   Soon after, the ADMINISTRATOR was ready to go.

5.12 Scenario 11 - Copy a File to the SYSVOL Share and Monitor SYSVOL Replication Between Domain Controllers

5.12.1 Scenario Overview

Total packets: 132

5.12.2 Procedure

1. A file was copied by USER1 from MACHINE1 to the SYSVOL share on DC1. For more information about SYSVOL, see [MS-ADTS].

Copyright © 2011 Microsoft Corporation.

2. The ADMINISTRATOR connected to the \\DC1\sysvol share and copied the a.txt file from the local machine to that share.

   Note This was straightforward SMB and took about 85 packets to fully complete.

3. Three seconds later, FRS began replication and the file was replicated to \\DC2\sysvol.

   Note This transaction totaled 30 packets.

5.13 Scenario 12 - Change Password

5.13.1 Scenario Overview

Total packets: 62

The purpose of this scenario was to change the ADMINISTRATOR's password on MACHINE1. Because this password is a domain password, MACHINE1 connected to a domain controller and the change was replicated to the DC.

5.13.2 Procedure

1. The ADMINISTRATOR pressed CTRL+ALT+DEL and selected the Change Password option.

   Note This procedure created a set of Kerberos transactions (with DC2, in this case) and was complete in 20 packets.

2. Domain replication took over and replicated the information to the other DC.

   Note This procedure was a short transaction that completed in 20 packets, primarily by using the RPC protocol as specified in [MS-DRSR].

5.14 Scenario 13 - Delete the Machine Account

5.14.1 Scenario Overview

Total packets: 892

This scenario took place logging on to DC1 in order to delete the MACHINE1 account. As with Scenario 1, the greatest part of this sniff was in opening the MMC.

5.14.2 Procedure

After opening the MMC, the ADMINISTRATOR opened the tree to view the machine accounts, selected the MACHINE1 account, and deleted it.

Note Deletion of the MACHINE1 account from the domain created a set of [LDAP] and SMB packets; then DCERPC traffic occurred between the domain controllers.

5.15 Scenario 14 - Promote a Server to First Domain Controller in a Domain

5.15.1 Scenario Overview

Total packets: 350

This scenario did not create a large amount of network traffic because most processing was done locally.

Copyright © 2011 Microsoft Corporation.

5.15.2 Procedure

1. Starting with a Microsoft Windows®–based server that was not joined to a domain, the DCPROMO utility was run to promote the server to a domain controller. This process also created the DNS server for this domain. There was some DNS traffic to query the existence of names before hitting the option to promote. During promotion of the server to the domain controller, some browser traffic and other related traffic occurred in order to elect the server to be the Domain Master Browser, as specified in [MS-BRWS] section 3.4.

   Note This was until packet 165 of the sniff.

2. After the server was promoted, the system was restarted and registration occurred for the relevant roles in DNS traffic.

   Note This was another 180 packets, primarily DNS and Common Internet File System (CIFS) Browser Protocol traffic.

3. Domain Machine and User accounts can now be created, as indicated in Scenarios 1 and 2.

5.16 Scenario 15 - Promote a Server to Replica Domain Controller

5.16.1 Scenario Overview

Total packets: 2623

This scenario is the largest and uses the most protocols of all the named scenarios.

5.16.2 Procedure

1. The ADMINISTRATOR logged on to a newly installed Microsoft Windows®–based server that had been joined to the domain. This created the traffic that is documented in Scenarios 1 and 3.

2. The ADMINISTRATOR then ran the DCPROMO utility to promote this server to a domain controller and selected the option to have this DC as part of an existing domain (that is, as part of the domain that was created in Scenario 14).

   Note This step was at 350 packets into the sniff.

3. After selecting the Next option, DCPROMO connected to the DC by using a combination of [LDAP], the Kerberos protocol, SMB, and the Directory Services Setup (DSSETUP) Remote Protocol packets in order to set the new DC. It then restarted the server to start up the server in its new mode.

   Note When tested, this restart occurred after 1,450 packets.

4. After the system was restarted, similar traffic occurred with the inclusion of SAM traffic. Eventually, the Windows Remote Registry Protocol values were written to the new DC.

   Note When tested, this step occurred at packets 1,970–2,100.

5. After the registry keys were transferred, similar traffic continued for another 200 packets and file replication started as part of the SYSVOL replication.

   Note This step started at about packet 2,300 and went until about packet 2,600.

6. The newly promoted domain controller was now fully replicated and began handling logon requests.

Copyright © 2011 Microsoft Corporation.

6 Appendix A: Join a Domain Scenario

6.1 Overview

A computer is considered joined to a domain when a certain state exists on the computer and also on the domain naming context of a domain controller. The state in the domain naming context is persisted on a computer account object as specified later in this section. The state enables the computer and the domain to authenticate by using various protocols, such as NetLogon, NTLM, and the Kerberos protocol.

There are several ways to initialize the state that is required to join a computer to a domain. The following example illustrates how a client computer that is running Windows® XP operating system and a Windows Server® 2003 operating system domain controller use certain protocols and sequences of events to accomplish the required state changes. In that example, the Security Account Manager (SAM) Remote Protocol (Client-to-Server) is used to create a computer account. However, an implementation could choose the LDAP protocol, as specified in [RFC2251], to accomplish the same.

6.2 Message Sequencing

The following diagram shows the sequence of steps that is performed by a client computer that is running Windows® XP operating system and by a Windows Server® 2003 operating system domain controller during the domain join operation.

Copyright © 2011 Microsoft Corporation.

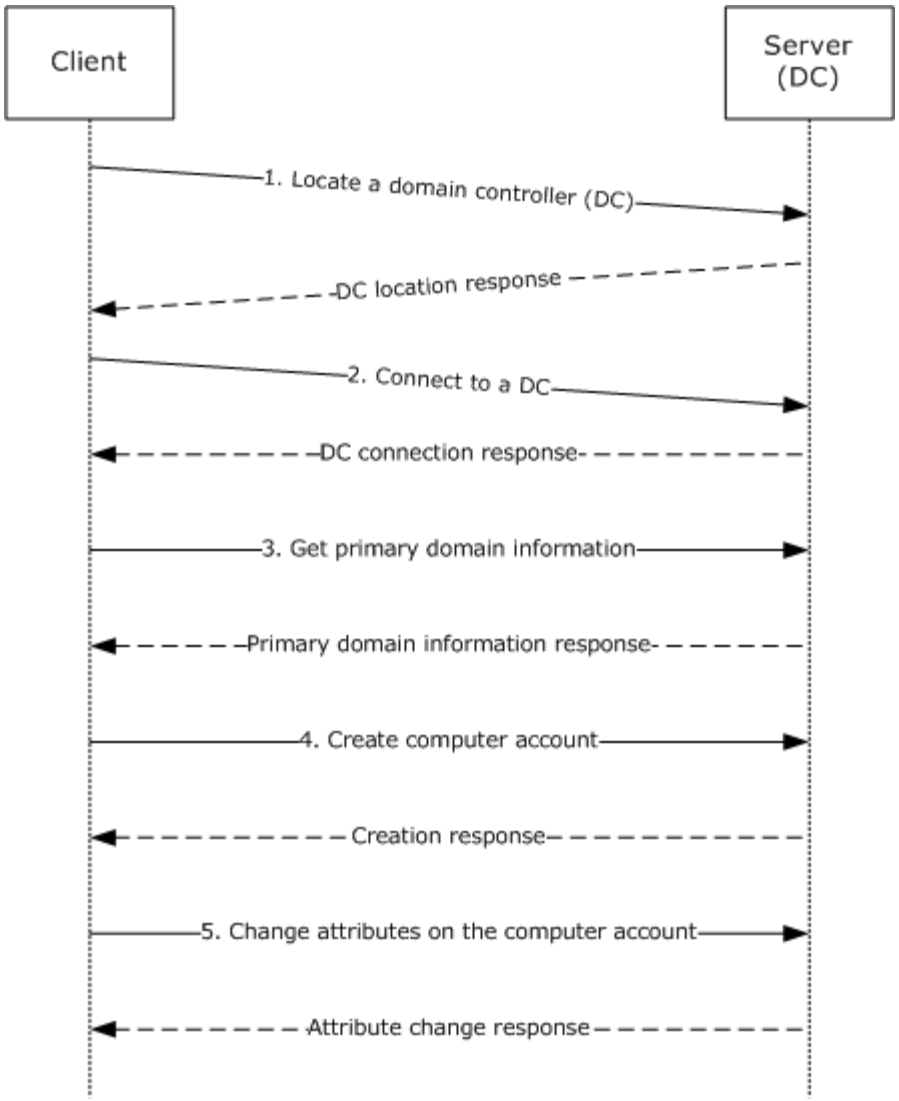


Figure 52: Message sequencing

6.3 State Table

The following state changes occur as part of this operation.

Initial State

Client	Server (DC)
PrimaryDomain.DomainName = "Workgroup"	PrimaryDomain.DomainName = "Test2"
PrimaryDomain.SID = NULL	PrimaryDomain.SID = {SID-Value}
Domain-Secret-Value = NULL	

Final State

Copyright © 2011 Microsoft Corporation.

Client	Server (DC)
PrimaryDomain.DomainName = "DomainA"	PrimaryDomain.DomainName = "Test2"
PrimaryDomain.SID = NULL	PrimaryDomain.SID = {SID-Value}
Domain-Secret-Value = {domain-secret}	A computer account object exists and it has the following LDAP attributes: ▪ userAccountControl = USER_WORKSTATION_TRUST_ACCOUNT ▪ samAccountName = "Machine2$" ▪ unicodePwd = {domain-secret} ▪ dNSHostName = "Machine2.Test2.com" ▪ servicePrincipalName = { "HOST/Machine2", "HOST/Machine2.Test2.com" }

6.4 Protocol Specification Reference Table

The following table lists the protocol documents that correspond to the operations shown in the preceding sequence diagram.

Protocol message	Protocol specification title	Details
1. Locate a domain controller	[MS-ADTS] Active Directory Technical Specification	Locating a domain controller, LDAP "ping"
2. Connect to a domain controller	[MS-SMB] Server Message Block (SMB) Protocol Specification	SMB_COM_NEGOTIATE SMB_COM_SESSION_SETUP_ANDX SMB_COM_TREE_CONNECT_ANDX
3. Get primary domain controller information	[MS-LSAD] Local Security Authority (Domain Policy) Remote Protocol Specification	LsarOpenPolicy2 LsarQueryInformationPolicy2 LsarQueryInformationPolicy
4. Create a computer account	[MS-SAMR] Security Account Manager (SAM) Remote Protocol Specification (Client-to-Server)	SamrConnect5 SamrEnumerateDomainsInSamServer SamrLookupDomainInSamServer SamrOpenDomain SamrCreateUser2InDomain SamrQueryInformationUser SamrGetUserDomainPasswordInformation SamrSetInformationUser2 SamrCloseHandle
5. Change attributes on the computer account	[RFC2251] Lightweight Directory Access Protocol (LDAP) [MS-DRSR] Directory Replication Service (DRS) Remote Protocol	IDL_DRSBind [MS-DRSR] IDL_DRSCrackNames [MS-DRSR] IDL_DRSUnbind [MS-DRSR]

Copyright © 2011 Microsoft Corporation.

Protocol message	Protocol specification title	Details
		ldap_bind_sW [LDAP] ldap_search_s [LDAP] ldap_modify_s [LDAP]

6.5 Sequencing Details

The following example illustrates the message sequence for computer Machine2 that wants to join a domain Test2 where DC-1 is the domain controller for Test2.

1. Locate a domain controller.

   The computer joining the domain (Machine2) locates the domain controller DC-1.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	DNS	DNS: QueryId = 0xA199, QUERY (Standard query), Query for _ldap._tcp.dc._msdcs.test2.com of type SRV on class Internet
DC-1.test2.com	Machine2	DNS	DNS: QueryId = 0xA199, QUERY (Standard query), Response - Success
Machine2	DC-1.test2.com	LDAP	LDAP: (CLDAP)Search Request, MessageID: 12, BaseObject: NULL, SearchScope: base Object, SearchAlias: neverDerefAliases
DC-1.test2.com	Machine2	LDAP	LDAP: (CLDAP)Search Result Entry, MessageID: 12, Status: Success
Machine2	DC-1.test2.com	LDAP	LDAP: (CLDAP)Search Request, MessageID: 13, BaseObject: NULL, SearchScope: base Object, SearchAlias: neverDerefAliases
DC-1.test2.com	Machine2	LDAP	LDAP: (CLDAP)Search Result Entry, MessageID: 13, Status: Success
Machine2	DC-1.test2.com	LDAP	LDAP: (CLDAP)Search Request, MessageID: 14, BaseObject: NULL, SearchScope: base Object, SearchAlias: neverDerefAliases
DC-1.test2.com	Machine2	LDAP	LDAP: (CLDAP)Search Result Entry, MessageID: 14, Status: Success
Machine2	192.168.20.255	NbtNs	NbtNs: Query Request for TEST2.COM <0x1C> Domain Controllers
Machine2	192.168.20.255	NbtNs	NbtNs: Query Request for TEST2.COM <0x1C> Domain Controllers
Machine2	192.168.20.255	NbtNs	NbtNs: Query Request for TEST2.COM <0x1C> Domain Controllers

2. Connect to a domain controller.

Copyright © 2011 Microsoft Corporation.

After locating a domain controller (DC-1), the computer joining the domain (Machine2) establishes an SMB session with the domain controller DC-1.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	TCP	TCP: Flags=.S....., SrcPort=1077, DstPort=Microsoft-DS(445), Len=0, Seq=541107809, Ack=0, Win=65535 (scale factor 0) = 65535
Machine2	DC-1.test2.com	TCP	TCP: Flags=.S....., SrcPort=1078, DstPort=NETBIOS Session Service(139), Len=0, Seq=757548711, Ack=0, Win=65535 (scale factor 0) = 65535
DC-1.test2.com	Machine2	TCP	TCP: Flags=.S..A..., SrcPort=Microsoft-DS(445), DstPort=1077, Len=0, Seq=4200896924, Ack=541107810, Win=16384 (scale factor 0) = 16384
Machine2	DC-1.test2.com	TCP	TCP: Flags=....A..., SrcPort=1077, DstPort=Microsoft-DS(445), Len=0, Seq=541107810, Ack=4200896925, Win=65535 (scale factor 0) = 65535
Machine2	DC-1.test2.com	SMB	SMB: C; Negotiate, Dialect = PC NETWORK PROGRAM 1.0, LANMAN1.0, Windows for Workgroups 3.1a, LM1.2X002, LANMAN2.1, NT LM 0.12
DC-1.test2.com	Machine2	TCP	TCP: Flags=.S..A..., SrcPort=NETBIOS Session Service(139), DstPort=1078, Len=0, Seq=1393655901, Ack=757548712, Win=16384 (scale factor 0) = 16384
Machine2	DC-1.test2.com	TCP	TCP: Flags=..R....., SrcPort=1078, DstPort=NETBIOS Session Service(139), Len=0, Seq=757548712, Ack=757548712, Win=0 (scale factor 0) = 0
DC-1.test2.com	Machine2	SMB	SMB: R; Negotiate, Dialect is NT LM 0.12 (#5)
Machine2	DC-1.test2.com	KerberosV5	KerberosV5: AS Request Cname: administrator Realm: test2.com Sname: krbtgt/test2.com
DC-1.test2.com	Machine2	KerberosV5	KerberosV5: AS Response Ticket[Realm: TEST2.COM, Sname: krbtgt/TEST2.COM]
Machine2	DC-1.test2.com	KerberosV5	KerberosV5: TGS Request Realm: TEST2.COM Sname: cifs/dc-1.test2.com
DC-1.test2.com	Machine2	KerberosV5	KerberosV5: TGS Response Cname: Administrator
Machine2	DC-1.test2.com	KerberosV5	KerberosV5: TGS Request Realm: TEST2.COM Sname: krbtgt/TEST2.COM
DC-	Machine2	KerberosV5	KerberosV5: TGS Response Cname: Administrator

Copyright © 2011 Microsoft Corporation.

Source	Destination	Protocol name	Description
1.test2.com			
Machine2	DC-1.test2.com	SMB	SMB: C; Session Setup Andx
Machine2	DC-1.test2.com	TCP	TCP: [Continuation to #114]Flags=...PA..., SrcPort=1077, DstPort=Microsoft-DS(445), Len=1190, Seq=541109407 - 541110597, Ack=4200897101, Win=65359 (scale factor 0) = 65359
DC-1.test2.com	Machine2	TCP	TCP: Flags=....A..., SrcPort=Microsoft-DS(445), DstPort=1077, Len=0, Seq=4200897101, Ack=541110597, Win=65535 (scale factor 0) = 65535
DC-1.test2.com	Machine2	SMB	SMB: R; Session Setup Andx
Machine2	DC-1.test2.com	SMB	SMB: C; Tree Connect Andx, Path = \\DC-1.TEST2.COM\IPC$, Service = ?????
DC-1.test2.com	Machine2	SMB	SMB: R; Tree Connect Andx, Service = IPC

3. Get primary domain controller information.

   The server then queries for the Domain Name and SID properties of "DC-1", and stores them locally.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	SMB	SMB: C; Nt Create Andx, FileName = \lsarpc
DC-1.test2.com	Machine2	SMB	SMB: R; Nt Create Andx, FID = 0x4000 (\lsarpc@#120)
Machine2	DC-1.test2.com	MSRPC	MSRPC: c/o Bind: UUID{12345778-1234-ABCD-EF00-0123456789AB} LSA Call=0x1 Assoc Grp=0x0 Xmit=0x10B8 Recv=0x10B8
DC-1.test2.com	Machine2	SMB	SMB: R; Write Andx, FID = 0x4000 (\lsarpc@#120), 72 bytes
Machine2	DC-1.test2.com	SMB	SMB: C; Read Andx, FID = 0x4000 (\lsarpc@#120), 1024 bytes at Offset 0
DC-1.test2.com	Machine2	MSRPC	MSRPC: c/o Bind Ack: Call=0x1 Assoc Grp=0x4076 Xmit=0x10B8 Recv=0x10B8
Machine2	DC-1.test2.com	LsaRpc	LsaRpc: LsarOpenPolicy2 Request, Target Computer: \\DC-1.test2.com, DesiredAccess: 0x02000000,
DC-	Machine2	LsaRpc	LsaRpc: LsarOpenPolicy2 Response, PolicyHandle:

Copyright © 2011 Microsoft Corporation.

Source	Destination	Protocol name	Description
1.test2.com			{00000000-CBB95A99-9210-9A4B-A14C-90288FE9E83C}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	LsaRpc	LsaRpc: LsarQueryInformationPolicy2 Request, InfoClass: PolicyDnsDomainInformation (0x0C) , Policy Handle: {00000000-CBB95A99-9210-9A4B-A14C-90288FE9E83C}
DC-1.test2.com	Machine2	LsaRpc	LsaRpc: LsarQueryInformationPolicy2 Response, PolicyDnsDomainInformation (0x0C) , Name: TEST2, DNSDomainName: test2.com, DNSForestName: test2.com, SID: S-1-000005-21-3120889152-234136572-295905738, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	LsaRpc	LsaRpc: LsarQueryInformationPolicy Request, InfoClass: PolicyPrimaryDomainInformation (0x03) , Policy Handle: {00000000-CBB95A99-9210-9A4B-A14C-90288FE9E83C}
DC-1.test2.com	Machine2	LsaRpc	LsaRpc: LsarQueryInformationPolicy Response, PolicyPrimaryDomainInformation (0x03) , PrimaryDomain: TEST2, SID: S-1-000005-21-3120889152-234136572-295905738, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	TCP	TCP: Flags=....A..., SrcPort=1077, DstPort=Microsoft-DS(445), Len=0, Seq=541111462, Ack=4200898344, Win=64116 (scale factor 0) = 64116

4. Create a computer account.

   Machine2 then creates a domain-object that sets the sAMAccountName, userAccountControl, and unicodePwd attributes.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	SMB	SMB: C; Nt Create Andx, FileName = \samr
DC-1.test2.com	Machine2	SMB	SMB: R; Nt Create Andx, FID = 0x0001 (\samr@#133)
Machine2	DC-1.test2.com	MSRPC	MSRPC: c/o Bind: UUID{12345778-1234-ABCD-EF00-0123456789AC} Security Account Manager Call=0x1 Assoc Grp=0x0 Xmit=0x10B8 Recv=0x10B8
DC-1.test2.com	Machine2	SMB	SMB: R; Write Andx, FID = 0x0001 (\samr@#133), 72 bytes
Machine2	DC-1.test2.com	SMB	SMB: C; Read Andx, FID = 0x0001 (\samr@#133), 1024 bytes at Offset 0
DC-	Machine2	MSRPC	MSRPC: c/o Bind Ack: Call=0x1 Assoc Grp=0x4077

Copyright © 2011 Microsoft Corporation.

Source	Destination	Protocol name	Description
1.test2.com			Xmit=0x10B8 Recv=0x10B8
Machine2	DC-1.test2.com	Samr	Samr: SamrConnect5 Request, ServerName: \\DC-1.test2.com, InVersion: 1, DesiredAccess: 0x00000030
DC-1.test2.com	Machine2	Samr	Samr: SamrConnect5 Response, OutVersion: 1, ServerHandle: {00000000-015F74FE-5669-FB44-AA4E-C3B5447F3991}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrEnumerateDomainsInSamServer Request, EnumerationContext: 0, PreferedMaxLength: 8192, ServerHandle: {00000000-015F74FE-5669-FB44-AA4E-C3B5447F3991}
DC-1.test2.com	Machine2	Samr	Samr: SamrEnumerateDomainsInSamServer Response, EnumerationContext: 4, Entries Returned: 2, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrLookupDomainInSamServer Request, DomainName: TEST2, ServerHandle: {00000000-015F74FE-5669-FB44-AA4E-C3B5447F3991}
DC-1.test2.com	Machine2	Samr	Samr: SamrLookupDomainInSamServer Response, DomainID: S-1-000005-21-3120889152-234136572-295905738, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrOpenDomain Request, DomainID: S-1-000005-21-3120889152-234136572-295905738, DesiredAccess: 0x00000211, ServerHandle: {00000000-015F74FE-5669-FB44-AA4E-C3B5447F3991}
DC-1.test2.com	Machine2	Samr	Samr: SamrOpenDomain Response, DomainHandle: {00000000-848748D8-18C0-244E-AE14-9570479F501E}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrCreateUser2InDomain Request, UserName: MACHINE2$, DesiredAccess: 0xE00500B0, AccountType: 0x00000080, DomainHandle: {00000000-848748D8-18C0-244E-AE14-9570479F501E}
DC-1.test2.com	Machine2	TCP	TCP: Flags=....A..., SrcPort=Microsoft-DS(445), DstPort=1077, Len=0, Seq=4200899206, Ack=541112595, Win=65029 (scale factor 0) = 65029
DC-1.test2.com	Machine2	Samr	Samr: SamrCreateUser2InDomain Response, GrantedAccess: 1, RelativeID: 1107, UserHandle: {00000000-A5D57417-6642-9F4B-B5CA-68BC62334840}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrQueryInformationUser Request, UserInformationClass: 0x0010 -

Copyright © 2011 Microsoft Corporation.

Source	Destination	Protocol name	Description
			UserControlInformation , UserHandle: {00000000-A5D57417-6642-9F4B-B5CA-68BC62334840}
DC-1.test2.com	Machine2	Samr	Samr: SamrQueryInformationUser Response, UserInformation: UserAccountControl: 0x00000085, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrGetUserDomainPasswordInformation Request, UserHandle: {00000000-A5D57417-6642-9F4B-B5CA-68BC62334840}
DC-1.test2.com	Machine2	Samr	Samr: SamrGetUserDomainPasswordInformation Response, PasswordInformation: MinPasswordLength: 0, PasswordProperties: 0x00000000, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrSetInformationUser2 Request, UserInformationClass: 0x0019 - UserInternal4InformationNew , UserInformation: , UserHandle: {00000000-A5D57417-6642-9F4B-B5CA-68BC62334840}
DC-1.test2.com	Machine2	Samr	Samr: SamrSetInformationUser2 Response, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrCloseHandle Request, SamHandle: {00000000-A5D57417-6642-9F4B-B5CA-68BC62334840}
DC-1.test2.com	Machine2	Samr	Samr: SamrCloseHandle Response, SamHandle: {00000000-00000000-0000-0000-0000-000000000000}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrCloseHandle Request, SamHandle: {00000000-848748D8-18C0-244E-AE14-9570479F501E}
DC-1.test2.com	Machine2	Samr	Samr: SamrCloseHandle Response, SamHandle: {00000000-00000000-0000-0000-0000-000000000000}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	Samr	Samr: SamrCloseHandle Request, SamHandle: {00000000-015F74FE-5669-FB44-AA4E-C3B5447F3991}
DC-1.test2.com	Machine2	Samr	Samr: SamrCloseHandle Response, SamHandle: {00000000-00000000-0000-0000-0000-000000000000}, Status = 0x00000000 - STATUS_SUCCESS
Machine2	DC-1.test2.com	SMB	SMB: C; Close, FID = 0x0001 (\samr@#133)
DC-1.test2.com	Machine2	SMB	SMB: R; Close, FID = 0x0001 (\samr@#133)

Copyright © 2011 Microsoft Corporation.

5. Change attributes on the computer account.

   The server then performs an LDAP bind operation, locates the computer object using IDL_DRSCrackNames and writes the dNSHostName and servicePrincipalName attributes.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	LDAP	LDAP: Bind Request, MessageID: 17, Version: 3
DC-1.test2.com	Machine2	LDAP	LDAP: Bind Response, MessageID: 17, Status: Success
Machine2	DC-1.test2.com	TCP	TCP: Flags=.S....., SrcPort=1085, DstPort=DCE endpoint resolution(135), Len=0, Seq=3567024150, Ack=0, Win=65535 (scale factor 0) = 65535
DC-1.test2.com	Machine2	TCP	TCP: Flags=.S..A..., SrcPort=DCE endpoint resolution(135), DstPort=1085, Len=0, Seq=1745756114, Ack=3567024151, Win=16384 (scale factor 0) = 16384
Machine2	DC-1.test2.com	TCP	TCP: Flags=....A..., SrcPort=1085, DstPort=DCE endpoint resolution(135), Len=0, Seq=3567024151, Ack=1745756115, Win=65535 (scale factor 0) = 65535
Machine2	DC-1.test2.com	MSRPC	MSRPC: c/o Bind: UUID{E1AF8308-5D1F-11C9-91A4-08002B14A0FA} Endpoint Mapper Call=0x1 Assoc Grp=0x0 Xmit=0x16D0 Recv=0x16D0
DC-1.test2.com	Machine2	MSRPC	MSRPC: c/o Bind Ack: Call=0x1 Assoc Grp=0x7AA1 Xmit=0x16D0 Recv=0x16D0
Machine2	DC-1.test2.com	EPM	EPM: Request: ept_map: NDR, DRSUAPI {E3514235-4B06-11D1-AB04-00C04FC2DCD2} v4.0, RPC v5, 0.0.0.0:135 (0x87) [DCE endpoint resolution(135)]
DC-1.test2.com	Machine2	EPM	EPM: Response: ept_map: NDR, DRSUAPI {E3514235-4B06-11D1-AB04-00C04FC2DCD2} v4.0, RPC v5, 192.168.20.1:1025 (0x401) [1025]
Machine2	DC-1.test2.com	TCP	TCP: Flags=.S....., SrcPort=1086, DstPort=1025, Len=0, Seq=2405308446, Ack=0, Win=65535 (scale factor 0) = 65535
DC-1.test2.com	Machine2	TCP	TCP: Flags=.S..A..., SrcPort=1025, DstPort=1086, Len=0, Seq=3080359050, Ack=2405308447, Win=16384 (scale factor 0) = 16384
Machine2	DC-1.test2.com	TCP	TCP: Flags=....A..., SrcPort=1086, DstPort=1025, Len=0, Seq=2405308447, Ack=3080359051, Win=65535 (scale factor 0) = 65535
Machine2	DC-1.test2.com	KerberosV5	KerberosV5: TGS Request Realm: TEST2.COM Sname: krbtgt/TEST2.COM
DC-1.test2.com	Machine2	KerberosV5	KerberosV5: TGS Response Cname: Administrator

Copyright © 2011 Microsoft Corporation.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	KerberosV5	KerberosV5: AP Request Ticket[Realm: TEST2.COM, Sname: ldap/dc-1.test2.com]
Machine2	DC-1.test2.com	TCP	TCP: [Continuation to #189]Flags=...PA..., SrcPort=1086, DstPort=1025, Len=1088, Seq=2405309907 - 2405310995, Ack=3080359051, Win=65535 (scale factor 0) = 65535
DC-1.test2.com	Machine2	TCP	TCP: Flags=....A..., SrcPort=1025, DstPort=1086, Len=0, Seq=3080359051, Ack=2405310995, Win=65535 (scale factor 0) = 65535
DC-1.test2.com	Machine2	KerberosV5	KerberosV5: AP Response
Machine2	DC-1.test2.com	MSRPC	MSRPC: c/o Alter Cont: UUID{E3514235-4B06-11D1-AB04-00C04FC2DCD2} DRSUAPI Call=0x1
DC-1.test2.com	Machine2	MSRPC	MSRPC: c/o Alter Cont Resp: Call=0x1 Assoc Grp=0x4078 Xmit=0x16D0 Recv=0x16D0
Machine2	DC-1.test2.com	TCP	TCP: Flags=....A..., SrcPort=1077, DstPort=Microsoft-DS(445), Len=0, Seq=541114196, Ack=4200899969, Win=65535 (scale factor 0) = 65535
Machine2	DC-1.test2.com	DrsUApi	DrsUApi: IDLDRSBind Request
DC-1.test2.com	Machine2	DrsUApi	DrsUApi: IDLDRSBind Response
Machine2	DC-1.test2.com	DrsUApi	DrsUApi: IDLDRSCrackNames Request
DC-1.test2.com	Machine2	DrsUApi	DrsUApi: IDLDRSCrackNames Response
Machine2	DC-1.test2.com	DrsUApi	DrsUApi: IDLDRSCrackNames Request
DC-1.test2.com	Machine2	DrsUApi	DrsUApi: IDLDRSCrackNames Response
Machine2	DC-1.test2.com	DrsUApi	DrsUApi: IDLDRSUnbind Request
DC-1.test2.com	Machine2	DrsUApi	DrsUApi: IDLDRSUnbind Response
Machine2	DC-1.test2.com	LDAP	LDAP: Search Request, MessageID: 18, BaseObject: CN=MACHINE2,CN=Computers,DC=test2,DC=com, SearchScope: base Object, SearchAlias: neverDerefAliases
DC-1.test2.com	Machine2	LDAP	LDAP: Search Result Entry, MessageID: 18, Status: Success

Copyright © 2011 Microsoft Corporation.

Source	Destination	Protocol name	Description
Machine2	DC-1.test2.com	LDAP	LDAP: Modify Request, MessageID: 19
DC-1.test2.com	Machine2	LDAP	LDAP: Modify Response, MessageID: 19, Status: Success
Machine2	DC-1.test2.com	LDAP	LDAP: Unbind Request, MessageID: 20

Copyright © 2011 Microsoft Corporation.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

Copyright © 2011 Microsoft Corporation.