[MS-AUTHSO]:
Windows Authentication Services System Overview

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 Open Specification Promise or the Community Promise. 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. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

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

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

Abstract

Describes a representative sample of authentication tasks that use the Windows Authentication Services System. Tasks described in this document incorporate protocols that in turn use authentication protocols, which include Interactive Domain Logon, Internet Web Access with HTTP, file services with SMB [MS-SMB] [MS-SMB2], and remote operations with RPC [MS-RPCE]. Negotiate [MS-SPNG], Kerberos [MS-KILE], NTLM [MS-NLMP], and APDS [MS-APDS] are some of the protocols that may be employed during authentication tasks.

Revision Summary

Date	Revision History	Revision Class	Comments
08/14/2009	0.1	Major	First Release.
09/25/2009	0.2	Minor	Updated the technical content.
11/06/2009	0.2.1	Editorial	Revised and edited the technical content.
12/18/2009	0.3	Minor	Updated the technical content.
01/29/2010	1.0	Major	Updated and revised the technical content.
03/12/2010	2.0	Major	Updated and revised the technical content.
04/23/2010	3.0	Major	Updated and revised the technical content.
06/04/2010	4.0	Major	Updated and revised the technical content.
07/16/2010	5.0	Major	Significantly changed the technical content.
08/27/2010	5.1	Minor	Clarified the meaning of the technical content.
10/08/2010	6.0	Major	Significantly changed the technical content.
11/19/2010	6.1	Minor	Clarified the meaning of the technical content.
01/07/2011	7.0	Major	Significantly changed the technical content.
02/11/2011	8.0	Major	Significantly changed the technical content.
03/25/2011	9.0	Major	Significantly changed the technical content.
05/06/2011	9.0	No change	No changes to the meaning, language, or formatting of

Date	Revision History	Revision Class	Comments
			the technical content.
06/17/2011	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
09/23/2011	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/30/2012	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	9.1	Minor	Clarified the meaning of the technical content.
01/31/2013	9.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

A "Defined Task" is a logical procedure that uses one or more Protocols or Systems to accomplish a specific goal. This Defined Task System Document describes four Tasks that are all related to the goal of authentication.

In conjunction with Protocol Technical Documents, which are primarily intended to cover Protocols, this Defined Task System Document presents and covers the rules for information exchange that are relevant to the Tasks, and the Protocols they use, that are used to interoperate or communicate with Windows client operating systems and selected Windows Server scenarios (those covered in published TDs) in its various environments.

The Windows Authentication Services System Overview covers the components necessary for authentication for the tasks of interactive logon, **HTTP** web access, file system services, and remote operations. Windows protocols may also provide their own authentication mechanisms as part of their design which is not part of the Windows Authentication Services System. Examples of this are:

- LDAP Simple Bind ([MS-ADTS] section 5.1.1.1.1):

  Simple Bind is an example of an authentication method which is part of the standard protocol.

- SASL Authentication ([MS-ADTS] section 5.1.1.1.2):

  SASL Authentication is an example of the LDAP protocol using the SASL authentication framework [RFC2829].

- Sicily Authentication ([MS-ADTS] section 5.1.1.1.3):

  Sicily is an example of an authentication method unique to the protocol.

Authentication is the action of proving identity to a network service or a resource provider. Authentication plays a central role in the Windows operating system as a basis for proof of identity and ability to control access through authorization. The authentication process can use services provided by clients, servers, and **domain controllers**. The servers and domain controllers work in conjunction to provide management of accounts; the client provides services to manage the credentials to be used to prove identity to the **server computer**. Domain controllers provide **directory services**, specifically **Active Directory**, which is the default Windows technology for storing identity information.

1.1 Glossary

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

Active Directory
directory
directory service (DS)
distinguished name (DN)
domain
domain account
domain controller (DC)
domain member (member machine)
Domain Name System (DNS)
domain user
fully qualified domain name (FQDN)

globally unique identifier (GUID)
group
Group Policy
HTTP client
HTTP server
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Key Distribution Center (KDC)
Lightweight Directory Access Protocol (LDAP)
NetBIOS
remote procedure call (RPC)
service principal name (SPN)
Server Message Block (SMB)
Simple and Protected GSS-API Negotiation Mechanism (SPNEGO)
site

The following terms are specific to this document:

client computer: The client role in the network topology of client/server/**domain controller**.

protocol server: The server role of a specific protocol.

server computer: The server role in the network topology of client/server/**domain controller**.

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

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

The following protocol abbreviations are used in this document:

CIFS: Common Internet File System

DFS: Distributed File System

HTTP: Hypertext Transfer Protocol

NTP: Network Time Protocol

1.2 References

1.2.1 Normative References

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

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

[MS-ADA3] Microsoft Corporation, "Active Directory Schema Attributes N-Z".

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

[MS-APDS] Microsoft Corporation, "Authentication Protocol Domain Support".

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

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

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

[MS-DPSP] Microsoft Corporation, "Digest Protocol Extensions".

[MS-DRSR] Microsoft Corporation, "Directory Replication Service (DRS) Remote Protocol".

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

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

[MS-N2HT] Microsoft Corporation, "Negotiate and Nego2 HTTP Authentication Protocol".

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

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

[MS-PAC] Microsoft Corporation, "Privilege Attribute Certificate Data Structure".

[MS-PKCA] Microsoft Corporation, "Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol".

[MS-RCMP] Microsoft Corporation, "Remote Certificate Mapping Protocol".

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

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

[MS-SAMR] Microsoft Corporation, "Security Account Manager (SAM) Remote Protocol (Client-to-Server)".

[MS-SFU] Microsoft Corporation, "Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol".

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

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

[MS-SPNG] Microsoft Corporation, "Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension".

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

[NEGOEX-DRAFT] Short, M., Zhu, L., Damour, K., and McPherson, D., "The Extended GSS-API Negotiation Mechanism (NEGOEX)", December 2010, http://tools.ietf.org/id/draft-zhu-negoex-02.txt

If you have any trouble finding [NEGOEX-DRAFT], please check here.

[PKU2U-DRAFT] Zhu, L., Altman, J., and Williams, N., "Public Key Cryptography Based User-to-User Authentication (PKU2U)", November 2008, http://tools.ietf.org/id/draft-zhu-pku2u-09.txt

If you have any trouble finding [PKU2U-DRAFT], please check here.

[Referrals-11] Raeburn, K., and Zhu, L., "Kerberos Principal Name Canonicalization and KDC-Generated Cross-Realm Referrals", July 2008, http://tools.ietf.org/internet-drafts/draft-ietf-krb-wg-kerberos-referrals-11

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

[RFC2136] Thomson, S., Rekhter Y. and Bound, J., "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997, http://www.ietf.org/rfc/rfc2136.txt

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

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., et al., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

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

[RFC2829] Wahl, M., Alvestrand, H., Hodges, J., and Morgan, R., "Authentication Methods for LDAP", RFC 2829, May 2000, http://www.ietf.org/rfc/rfc2829.txt

[RFC2831] Leach, P., and Newman, C., "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000, http://www.ietf.org/rfc/rfc2831.txt

[RFC3244] Swift, M., Trostle, J., and Brezak, J., "Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols", RFC 3244, February 2002, http://www.ietf.org/rfc/rfc3244.txt

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005, http://www.ietf.org/rfc/rfc3961.txt

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

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

[RFC4556] Zhu, L., and Tung, B., "Public Key Cryptography for Initial Authentication in Kerberos", RFC 4556, June 2006 http://www.ietf.org/rfc/rfc4556.txt

[RFC4559] Jaganathan, K., Zhu, L., and Brezak, J., "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006, http://www.ietf.org/rfc/rfc4559.txt

[RFC5349] Zhu, L., Jaganathan, K., and Lauter, K., "Elliptic Curve Cryptography (ECC) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 5349, September 2008, http://www.ietf.org/rfc/rfc5349.txt

1.2.2 Informative References

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

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

2 Overview

Section 1, "Introduction", primarily describes this Defined Task System Document per se. This section introduces the Tasks that are being documented.

All of the tasks in this document use a number of common authentication protocols that are surfaced to the operating system and applications via standard APIs. These protocols enable authentication of users, computers, and services.

2.1 Summary

Authentication is an essential operation in providing services and resources in a networked environment. Typically, identity is proven by a cryptographic operation that uses either a key only the user knows, as with public key cryptography, or a shared secret key, as with a password. The server side of the authentication exchange compares the signed data with a known cryptographic key to validate the authentication attempt.

Authentication techniques range from a simple logon, which identifies users based on something that only the user knows, such as a password, to more powerful security mechanisms that use something that the user has, such as tokens and smart cards (public key certificates). In a business environment, users might access multiple applications on many types of servers within a single location or across multiple locations via a network. For these reasons, authentication is supported from within the operating system.

2.2 List of Tasks

The Windows Authentication Services System tasks described in this document are as follows:

Interactive Domain Logon Task: This task describes authentication steps that are taken during an interactive logon to a **client computer** joined to a **domain**.

HTTP Access Authentication - Server Task: This task describes the server-side authentication steps that are taken during processing of an **HTTP** request for a protected resource.

File System Services Authentication - SMB Server Task: This task describes the server-side authentication steps that are taken during processing of an **SMB** connection.

Remote Procedure Services Authentication - RPC Server Task: This task describes the server-side authentication steps that are taken during processing of an **RPC** request.

2.3 Relevant Standards

There are several standards used by these four authentication tasks. The following diagram shows these standards and their relationship to the **domain members**, servers, and domain controllers that host the domain **directory** database.

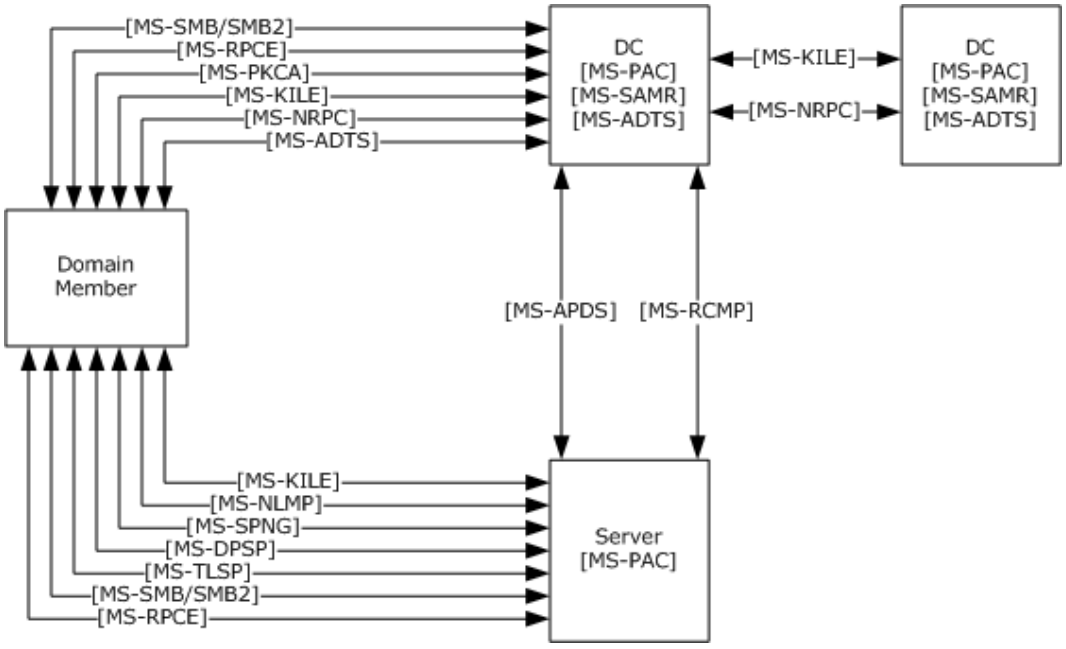

Figure 1: Standards used by authentication tasks

Protocols that are directly used or relevant to tasks described in this document are as follows:

Lightweight Directory Access Protocol (LDAP), as specified in [MS-ADTS]. **LDAP** is the primary method for accessing and manipulating objects in a directory service such as Active Directory.

Security Accounts Manager (SAM) RPC, as specified in [MS-SAMR]. The SAM RPC interface is an alternate method of manipulating a subset of objects in the directory, and a subset of the properties associated with account objects in an Active Directory-style directory. It is still frequently used, although LDAP is a richer interface.

Kerberos ([MS-KILE], [RFC4120], [MS-PAC], and [RFC3244]). Kerberos is the primary authentication protocol for Active Directory-style domains. The Kerberos **Key Distribution Center (KDC)** **authenticates** principals based on the Kerberos protocol, using the directory as the back-end store for account information.

Netlogon RPC [MS-NRPC]. Netlogon is the primary protocol used between members of a domain and the domain controller for non-Kerberos authentication. Netlogon RPC serves as the general channel by which a client of the domain connects to a domain controller for the purposes of authentication, authorization, or related security purposes.

- Protocols from [MS-APDS] that use the Netlogon RPC interface above for authentication validation and authorization:
  - Kerberos PAC validation [MS-PAC]. The Privilege Attribute Certificate or PAC that is carried in the Kerberos tickets is documented in [MS-PAC]. The PAC is digitally signed by the issuing KDC using a method described in [MS-PAC] section 2.8.1. A domain client can validate the signature by forwarding the PAC to the domain controller over this Netlogon RPC channel.
  - Digest validation ([RFC2617], [RFC2831], [MS-DPSP]). Digest authentication is an authentication protocol used primarily in HTTP authentication, but occasionally in LDAP-accessible directories as well. Digest authentication is similar to the NTLM [MS-NLMP]

protocol, and validation can take place at the domain controller instead of on the domain member that is handling the server end of the Digest authentication. In much the same way as NTLM over Netlogon is handled, the Digest authentication can be forwarded to the domain controller over the Netlogon channel.

  - NTLM/Netlogon [MS-NRPC] [MS-NLMP]. NTLM is the secondary authentication protocol for Active Directory-style domains (Kerberos is the primary) and the only authentication protocol for Windows NT operating system 4.0-style domains. NTLM authentication is forwarded to a domain controller over the Netlogon RPC interface, allowing servers to authenticate clients without having a complete account store local to the server. As with the Kerberos protocol, Active Directory is used as the account store for authentication.

- Remote Certificate Mapping Protocol/Netlogon [MS-RCMP]. Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are authentication protocols based on the use of X.509 public key certificates. The association between the certificate and an account is done via the Certificate Mapping Protocol, which travels over the Netlogon RPC protocol. Active Directory is used as the account store.

SMB/CIFS [MS-SMB] [MS-SMB2]. SMB/CIFS is used for access to the domain controller and servers for file and print operations and for certain RPC operations.

RPC [MS-RPCE]. RPC is used to provide interprocess communications for distributed applications.

Additionally, authentication tasks make use of typical network infrastructure protocols such as TCP, UDP, **DNS**, DHCP, and domain interactions [MS-DISO].

3 Background Knowledge and System-Specific Concepts

This section identifies the theoretical and practical information needed to understand this document and the Tasks in this System, and summarizes:

- Background knowledge that is required to understand this document.
- Concepts that are specific to the Tasks in this System.

An understanding of Windows domains and client/server topologies is required. Basic networking knowledge including DNS, DDNS, DHCP, TCP, and IP is required.

Basic knowledge of each authentication protocol including Digest, Kerberos, NTLM, and TLS is required. The Netlogon RPC interface provides a secure channel for other authentication protocols to execute their authentication, authorization, or other activities at the domain controller.

The following diagram shows the NTLM and Netlogon stack on the server and DC, which is used for NTLM authentication.

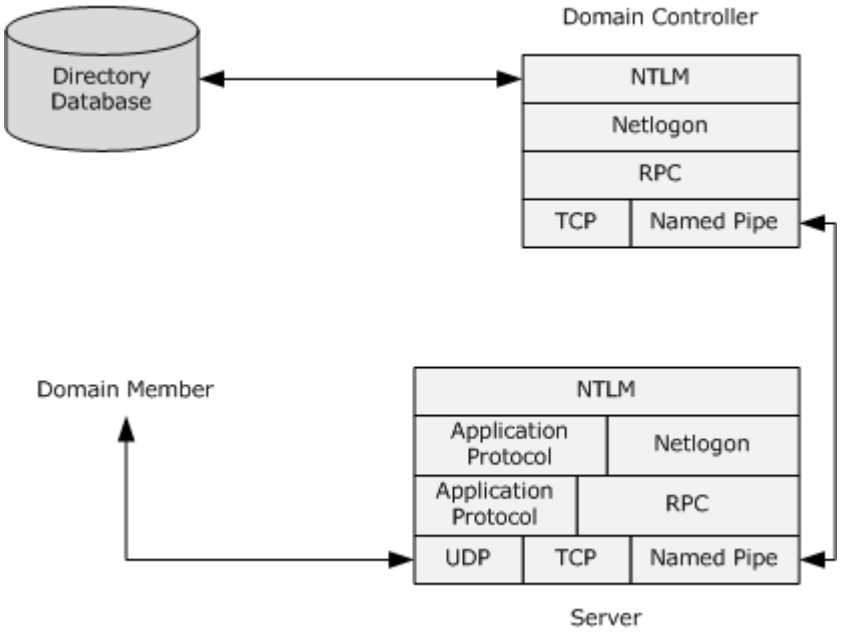

Figure 2: NTLM authentication stack

The following diagram shows the Digest and Netlogon stack on the server and DC, which is used for Digest authentication.

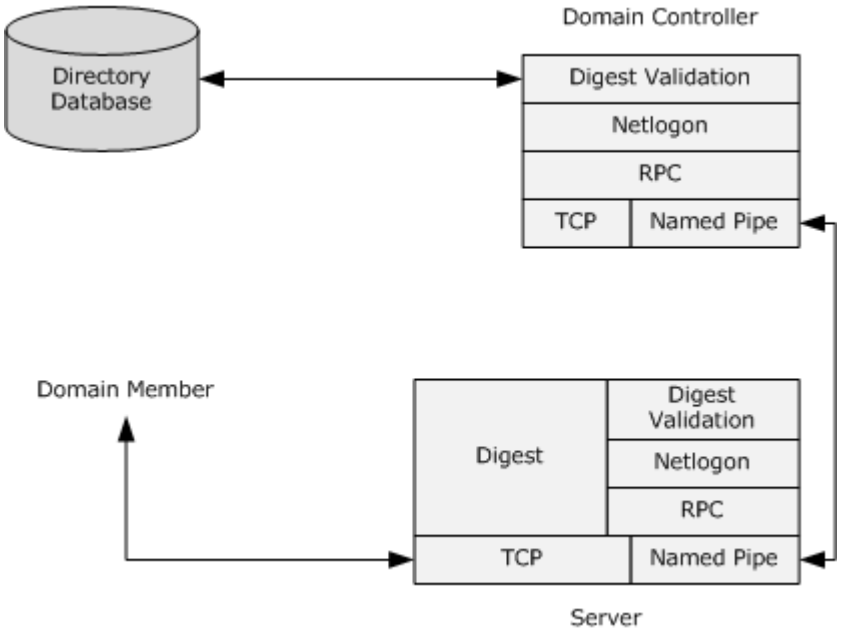

Figure 3: Digest authentication stack

Knowledge of encryption and data hashing is recommended.

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

The following diagram shows the TLS and Netlogon stack on the server and DC, which is used for TLS authentication and certificate mapping.

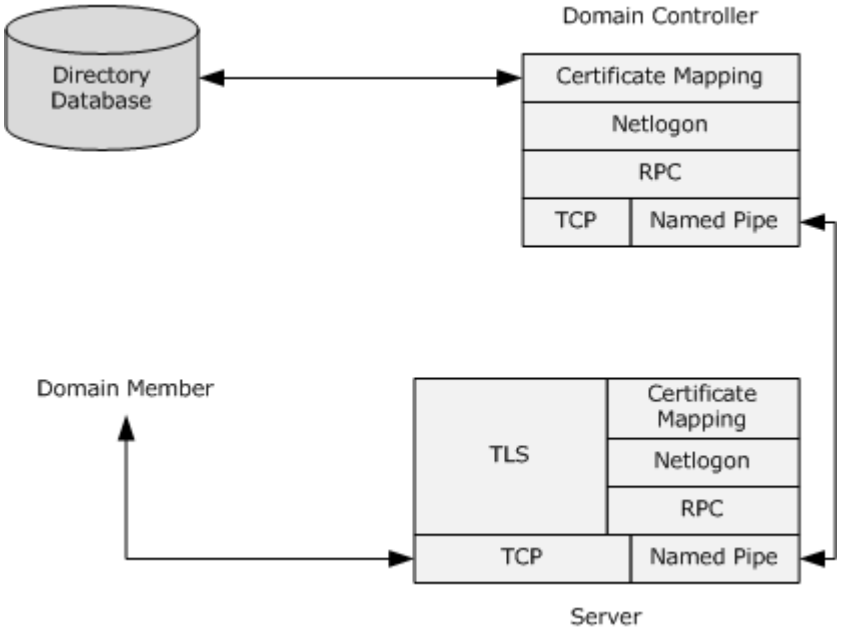

Figure 4: TLS authentication stack

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

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

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

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

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

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

3.2.1 Description and Flow

SPNEGO fits into the system as a security protocol and shuttles the actual messages of the underlying security protocols back and forth between the client and the server. SPNEGO does not implement any specific security features but performs the important operation of enforcing policy among security protocols.

SPNEGO is defined as a GSS mechanism, which follows the general model of passing security tokens from one machine to another. SPNEGO is a simple protocol that passes a list of acceptable security protocols (also called mechanisms) along with an optimistic mechanism token from one of these protocols. An optimistic mechanism token is the first context establishment token of the initiator's preferred mechanism embedded in the initial negotiation message, as specified in [RFC4178] section 4.2. The response from the server is what protocol has been selected and, optionally (if it matched the optimistic mechanism token), a response from that security protocol.

The SPNEGO module asks each available system protocol, in turn, if that protocol can potentially authenticate the server that is named by the component invoking SPNEGO. In the Windows implementation, the first protocol to respond positively is selected as the optimistic choice. After the protocol is selected, SPNEGO gets out of the way and lets the application protocol work directly with the selected authentication protocol.

3.2.2 Notes

In Windows, SPNEGO has the important task of deciding which protocol is valid for a particular connection. It does this by interpreting a number of input states and then adjusting the list of

security protocols it offers the server. The SPNEGO implementation knows that NTLM cannot implement the same guarantees (due to legacy) that more modern protocols can. Therefore, **NTLM** is treated in a special manner on a Windows-based system. To decide the most appropriate protocol, the SPNEGO implementation takes the following information:

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

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

The Microsoft Windows Negotiate Security Support Provider (SSP) InitializeSecurityContext (ISC) and AcceptSecurityContext (ASC) APIs allow applications to pass an NTLM message or **Kerberos** message as an input parameter. These APIs identify this parameter and route it appropriately to the NTLM or Kerberos SSP. With this functionality available, applications can choose to use the negotiate ISC/ASC to negotiate available packages or pass an NTLM or Kerberos message select NTLM [MS-NLMP]or Kerberos [MS-KILE] authentication. This behavior is known as the Universal Receiver behavior and is not required for authentication protocol compatibility.

3.3 SPNEGO Extended Negotiation (NEGOEX) Security Mechanism

SPNEGO Extended Negotiation (NEGOEX) Security Mechanism [NEGOEX-DRAFT] enhances the capabilities of **SPNEGO** [RFC4178] by providing a security mechanism which can be negotiated by SPNEGO. When the NEGOEX security mechanism is selected by SPNEGO, NEGOEX provides a method allowing selection of a common authentication protocol based on factors beyond just the fact that both client and server support a given security mechanism. NEGOEX provides support for authentication protocols to exchange additional information to determine whether the protocol is a usable common mechanism for both the client and server. For example, an authentication protocol might be supported by both the client and server, but they might not have a trusted certificate authority in common. When using SPNEGO, this protocol could be selected but then fail later in the protocol when validating a certificate. With NEGOEX, the authentication protocol could exchange trust information and determine that there is no shared trust prior to being selected. This would then allow another security mechanism to be attempted.

In Windows, NEGOEX negotiates the following security mechanism:

- Public Key Cryptography Based User-to-User Authentication (PKU2U) [PKU2U-DRAFT]. The **globally unique identifier (GUID)** used for PKU2U is 235F69AD-73FB-4dbc-8203-0629E739339B.

3.4 Kerberos

The following diagram shows the Kerberos and Netlogon stack on the server and DC, which is used for Kerberos authentication and PAC validation for authorization.

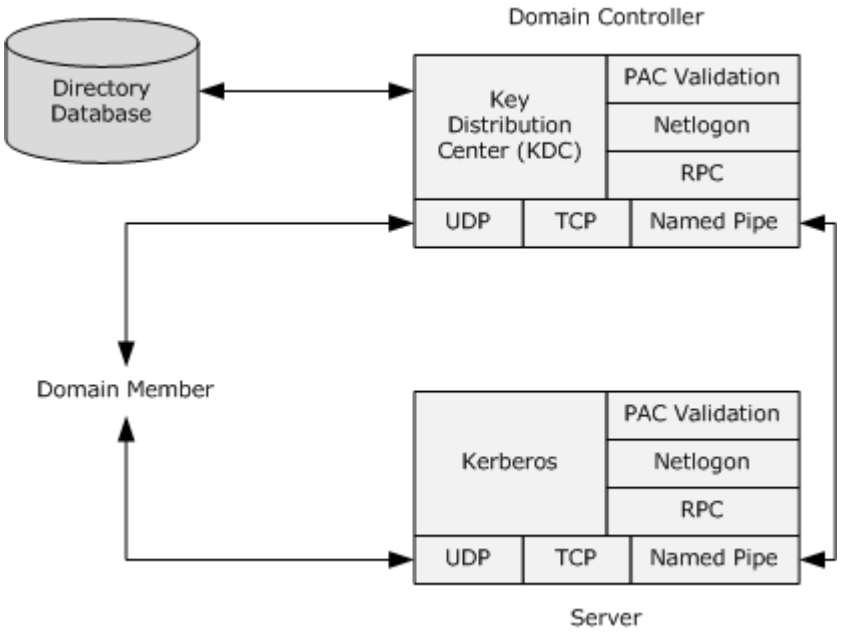

Figure 5: Kerberos authentication stack

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

3.4.1 Description and Flow

The Kerberos protocol is based on a client and a server and a trusted third party called the Key Distribution Center (KDC). The KDC is associated with an account database and has a key that is shared with each client or server that it knows about. The management of the account database is explicitly outside the Kerberos protocol. It is presumed, however, that each client or server knows its own key through some method.

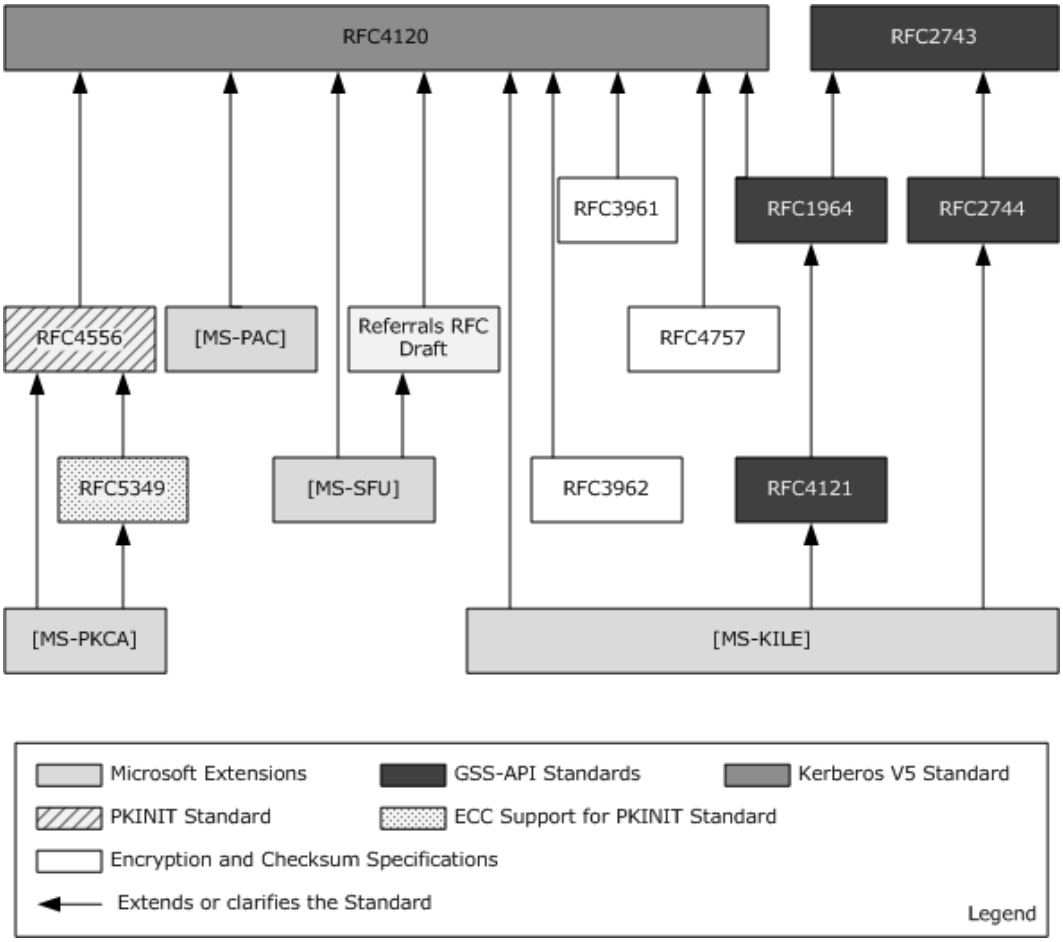

Figure 6: Relationships between Kerberos protocol standards and Microsoft extensions

For more detailed information, see the RFCs and corresponding technical documents:

- [RFC4120] "The Kerberos Network Authentication Service (V5)".
- [MS-KILE] "Kerberos Protocol Extensions", which:
  - Specifies Microsoft choices for SHOULDs, MAYs, and options in [RFC4120] and [RFC3961], and clarifies behavior left to the implementer
  - Extends the GSS-API RFCs with two new APIs ([MS-KILE] section 3.4.5)
  - Extends [RFC4120] with
    - New pre-authentication data by using the RFC's extensibility point ([MS-KILE] section 3.1.5)
    - New elements by using the RFC's optional authorization data elements ([MS-KILE] section 2.2)
    - New KRB-ERROR clock skew data ([MS-KILE] section 2.2.1)
    - Support for using the AD as the Kerberos account database ([MS-KILE] section 3.3.1.1)

        - Support for Windows authorization data by:
            - Generating a PAC ([MS-KILE] section 3.3.5.6.2)
            - Adding domain local group membership to a received PAC ([MS-KILE] section 3.3.5.7.3)
- [MS-SFU] "Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol Specification", which:
    - Extends [RFC4120] with:
        - Support for Service-for-User-to-Self
        - Support for Service-for-User-to-Proxy
        - Support for tracking services that have been delegated, by adding a new structure in the PAC.
- [RFC4556] "Public Key Cryptography for Initial Authentication in Kerberos"
- [MS-PKCA] "Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol Specification", which
    - Specifies Microsoft choices for SHOULDs and MAYs, and options in [RFC4556] and [RFC5349]
    - Normatively documents behavior from an earlier draft of [RFC4556]
- [Referrals-11] "Kerberos Principal Name Canonicalization and KDC-Generated Cross-Realm Referrals"

3.4.2 Notes

The Windows implementation uses **service principal names (SPNs)** to identify servers in TGS-REQs. Each server owning an SPN is responsible for ensuring that its SPN is in the **servicePrincipalName** attribute (as specified in [MS-ADA3] section 2.253) of its account in the AD. Windows has a "RestrictedKrbHost" *serviceclass* (as specified in [MS-KILE] section 3.1.5.11) which can be used by applications when their callers have provided the hostname but not the correct SPN for the server; which causes Kerberos to be used. This special *serviceclass* ([MS-KILE] section 3.1.5.11) has security considerations ([MS-KILE] section 3.1.2) and is never used by Windows applications using GSS-API.

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

Windows uses this *auth_data* field in the ticket to pass along the SID of the account and to group SIDs during authentication. The structure that is used for this behavior is termed the privilege attribute certificate (PAC), as specified in [MS-PAC]. The PAC contains all the account's group memberships and is used to provide sufficient context on the server to make authorization decisions (for example, "can this user open this file?"). Windows also includes additional profile and account management information, such as the location of the account's root directory in the PAC (as specified in [MS-PAC] section 2.5). This design greatly aids certain enterprise deployment scenarios. This additional information is used for supporting Windows interactive logon.

When the ticket includes authorization data that is respected and interpreted by something other than code running as that user, the potential for threat arises. In the Windows implementation, the

authorization data results in a Windows *access token*, which is a system-provided object that encapsulates an account's identity, group memberships, and system privileges. An access token on Windows is used to make authorization decisions by the system (for example, validating access to a file).

A malicious user might construct a ticket to itself that contains a PAC indicating that the user is a member of a group (for example, the Administrators group) that it should not be. If processed naively, the system accepts this PAC and allows the user inappropriate access to the system. Windows compensates for this by involving the DC when a nonprivileged server receives a ticket. The Windows Kerberos implementation uses Kerberos PAC validation (as specified in [MS-APDS] section 3.2) to verify the contents of the PAC.

3.5 Authorization

After an identity is suitably authenticated, the natural next step is to use that identity to authorize access to a resource. Windows has a very expressive authorization model available for applications and system components to use for making authorization decisions. See [MS-WSO] section 3.1.2.3 for an overview of Windows Authorization and related Security Descriptors. See [MS-DTYP] section 2.4 for definitions of constructed security types.

Windows was originally designed to meet the requirements of the C2 level of the Trusted Computer System Evaluation Criteria (TCSEC). The TCSEC program has since been supplanted by profiles written under the Common Criteria for Information Technology Security Evaluation, such as the Controlled Access Protection Profile. These profiles and related information can be found in [MS-WSO] section 1.2.

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

4 Interactive Domain Logon Task

This section describes the Interactive Domain Logon task. This task is expected to be used for user access to both local and domain resources. To use a computer running the Windows operating system, it is required to use system services. To use a system service, the account requesting the service will be a client of the service, and all clients will need to be authenticated by the service. Authentication requires the service to have validated proof of the user's credentials. The Interactive Domain Logon task begins when a user enters credentials to log on using the Windows user interface. The credentials consist of a username and password for local logons, and the user's username, password, and domain for domain logons. Kerberos PKINIT can also be used once the user unlocks a smart card using his or her personal identification number (PIN).

Users can perform an interactive logon by using a local user account (for local logon) or a **domain account** (for domain logon). The interactive logon process confirms the user's identification to a security database on the user's local computer or to a domain's directory service. This mandatory logon process cannot be turned off for users in a domain.

Users can perform an interactive logon to a computer in either of two ways:

- Locally, when the user has direct physical access to the computer.
- Remotely, through Terminal Services, in which case the logon is further qualified as remote interactive.

After an interactive logon, Windows runs applications on the user's behalf, and the user can interact with those applications to access resources (either locally or on remote computers) that were access-controlled to allow that user.

A local logon grants a user access to Windows resources on the local computer. A local logon requires that the user have a user account in the Security Accounts Manager (SAM) on the local computer. The SAM protects and manages user and **group** information in the form of security accounts stored in the local computer registry. The computer can have network access, but it is not required. Local user account and group membership information is used to manage access to local resources.

A domain logon grants a user access to local and domain networked resources. A domain logon requires that the user have a user account in Active Directory. The computer MUST have an account in Active Directory and be physically connected to the network. Users are configured to have the user rights to log on to a local computer or the domain. **Domain user** account information and group membership information is used to manage access to domain and local resources.

For Interactive Domain Logon, the validation will be focused on authenticating domain user credentials against the domain's directory service. The documentation that is provided here for the Interactive Domain Logon task describes how various protocols work in conjunction to accomplish the task.

4.1 Task Overview

4.1.1 Task Purpose

Windows operating system users are required to be authorized to use system services. This task is expected to be used for domain user access to local and domain network resources.

4.1.2 Task Applicability

This task is used when a user requires access to local and domain network resources.

4.1.3 Task Use Cases

4.1.3.1 Stakeholders and Interests Summary

The stakeholders for this task are:

Client User/User: The Client User is the end user of the system, accessing resources within the domain, or through a domain member workstation or computer. For this task, this entity is the primary stakeholder and initiates the task action. The task is being performed at the request of the User to gain access to local and network resources within the domain.

Client Computer: The client computer is the consumer of the services offered by the server. The client computer relies upon the domain to possibly locate or otherwise rendezvous with the server, authenticate the client computer to the server (and possibly also the server to the client computer), and provide the server with whatever information is necessary for the client to receive the services from the server. Specifically for this task, the client computer to be used for the interactive logon MUST be a domain-joined member machine (a client computer, server computer, or even a domain controller).

4.1.3.2 Supporting Actors and Task Interests Summary

The supporting actors for this task are:

Domain Administrator: The Domain Administrator controls the domain. The Domain Administrator and the server administrator (described in section 5.1.3.2) need not be the same person, and in many larger enterprises, they are not. The Domain Administrator controls the domain and all principals within the domain. The Domain Administrator relies upon the domain to provide the basis for authentication for principals communicating on the network, to provide the necessary authorization information to enforce policies across the domain and on specific servers, and to distribute policy controlling certain behavior to some or all principals of the domain. For this task, this entity provisions the domain credentials that allow the user to gain access to network resources.

Domain Controller: The Domain Controller provides authentication of the credentials and user-specific information supplied by the Domain Administrator and grants access by providing data necessary for evaluation in the authorization check.

4.1.3.3 Use Case Diagrams

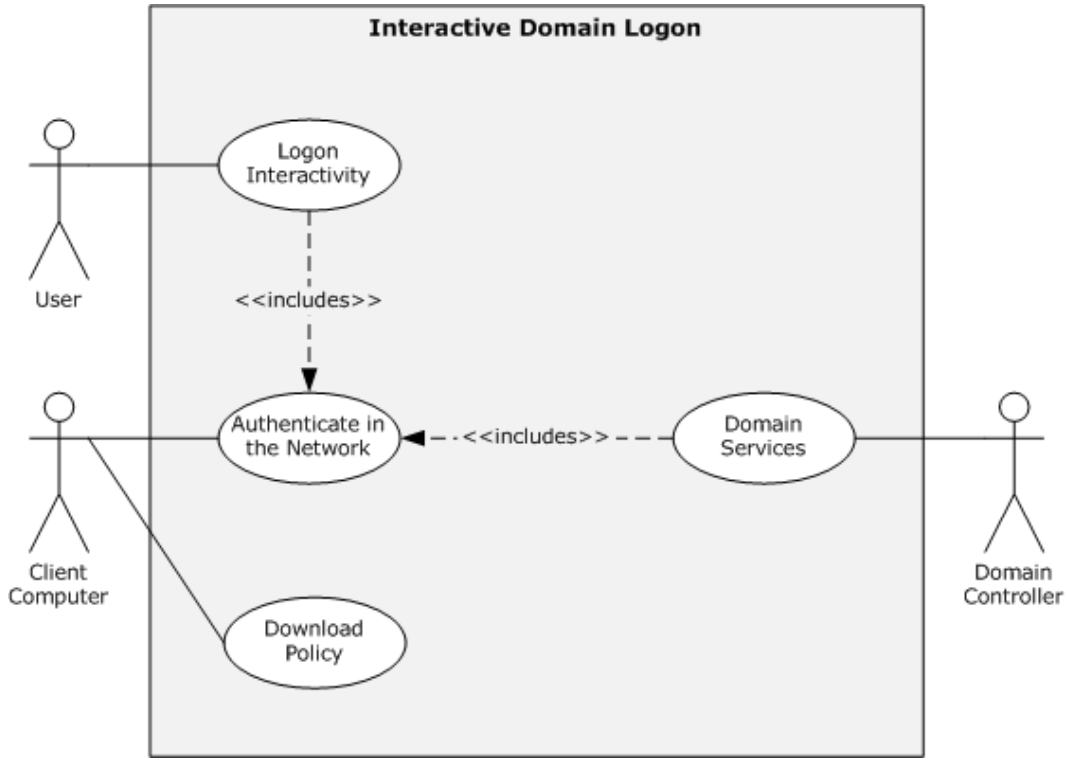

Figure 7: Interactive Domain Logon use case diagram

In this use case diagram the user is initiating the Interactive Domain Logon. The client computer has been successfully authenticated and joined to the domain infrastructure that is provided by the domain controller. The member machine has downloaded domain-specific policy. After the user initiates Interactive Domain Logon, authentication of the user's credentials takes place and, upon success, user information is supplied.

4.1.3.4 Interactive Domain Logon - User

For the use case of Interactive Domain Logon, the user is initiating the task with the goal of access to local and network resources authorized to the user. The user will provide credential material to initiate this action, which will provide identification and proof of that identification. The domain controller authenticates the user credentials and, if authentication is successful and the user is authorized, user interactive domain logon will be permitted.

Goal: The user needs to access local and network resources authorized to the user.

Context of Use: This is typically done when the user logs on to the computer.

Direct Actor: The user is the direct actor.

Primary Actor: Same as the direct actor.

Supporting Actors: The domain controller is the supporting actor.

Stakeholders and Interests:

- The user who is attempting to access local and network resources.
- The domain controller that is available to authenticate the user.

Preconditions:

- The domain controller is available.
- The user account is provisioned.

Minimal Guarantees: The user will be authenticated or the authentication will fail.

Success Guarantee: The user is authenticated.

Trigger: The user attempts to log on to the computer.

Main Success Scenario:

1. The user enters a username and password.
2. The Client Computer sends an authentication request to the Domain Controller.
3. The Domain Controller authenticates the user.
4. The Client Computer confirms user authorization for local resources.
5. The Client Computer creates a user environment.

Extensions: The User uses a smartcard and PIN.

4.2 Task Context

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

The Interactive Domain Logon task is a primary initial task for a user to accomplish in order to gain access to network resources within a Windows domain.

4.2.1 Task Environment

This task makes use of several protocols that exchange messages between the client machine and a domain controller. Many of the protocols have been previously used when the computer joined the domain administered by the domain controller.

4.2.2 Task Relationships

4.2.2.1 Black Box Relationship Diagrams

The task of Interactive Domain Logon uses a number of protocols (some are prerequisite and others are used within the Task itself). The User initiates the task. The task involves a member machine joined to the domain as well as the authority for the domain, that is, the Domain Controller.

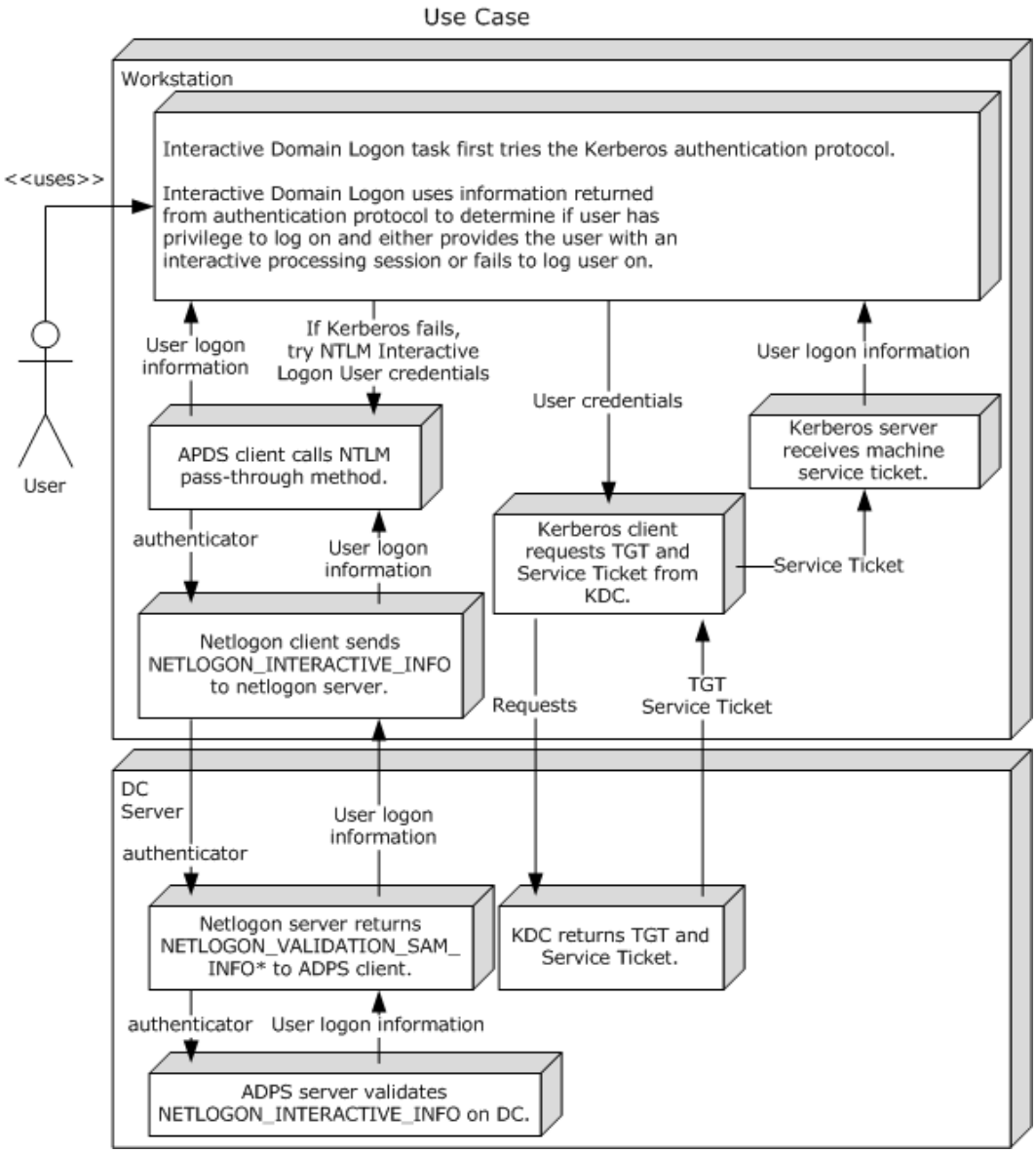

Figure 8: Black box relationships for the Interactive Domain Logon Task

4.2.2.2 Task Dependencies

This task has a number of external dependencies, described as follows:

- Precondition - System Level
  - Domain Interaction System Overview [MS-DISO]
  - PKI

The following protocols are used by this task:

- Precondition - Protocols
  - TCP/IP
  - DNS
  - LDAP
  - RPC
  - SMB
  - NetLogon
  - Kerberos
  - Authentication Protocol Domain Support [MS-APDS]
  - DFS
  - **Group Policy**
  - **NetBIOS**
  - NTP

The following tasks are used by this task:

- Precondition - Tasks
  - Domain Join

The following external server systems are used by this task:

- External Server System
  - Active Directory
  - DNS Directory
  - LDAP Directory
  - KDC
  - NTP Server
  - SMB Server

4.2.2.3 Task Influences

The Interactive Domain Logon task has several external influences. These influences can alter state such as logging the last authentication attempt or account lockout (that is, blocking a password guessing attempt) on the external server systems listed in section 4.2.2.2.

The external influences are as follows:

- Group Policy for the computer: provides configuration information and settings that may influence the operation of the computer.

- Group Policy for the user account for the domain logon: provides configuration information and settings that may influence how the user environment operates.
- Active Directory - account lockout subsystem: provides configurable security controls to block authentication based on repeated failed authentications.
- Security event logging on the domain controller: provides management and reporting of events related to the security environment of authentication actions.
- Event logging on NTP server: event logging may be enabled on the NTP server used by the client.

4.2.3 Task Assumptions and Preconditions

This task has the precondition that the domain controller has been set up and configured to support the domain infrastructure. The user account for the client member logon has been created and provisioned on the domain controller by the domain administrator.

Also, as part of the Windows startup procedure for the client computer, the following actions have taken place (for additional details see "domain join" in [MS-DISO] and section 4.4.1 in this document):

- Successful connection to network and startup of networking stacks.
- Network discovery of **site** and domain controller.
- Establishment of secure channel with domain controller.
- Download of Group Policy.
- Client certificate auto-enrollment.
- Time synchronization.
- Dynamic DNS update.

4.2.4 Task Versioning and Capability Negotiation

Beginning with Windows 2000 operating system, clients first try using the Kerberos protocol to authenticate for this task as described in section 4.4.5. Prior to Windows 2000, clients only used NTLM Interactive Logon ([MS-APDS] section 3.1.5.1).

The SMB protocol performs negotiation to select between SMB [MS-SMB] or SMB2 [MS-SMB2].

4.3 Task Architecture

The basic architecture contains a domain controller and a member machine. The member machine may be any member client computer or member server computer. The user is initiating an Interactive Domain Logon to the member machine.

4.3.1 Task Architectural Constraints

The primary constraint is that the computer has successfully accomplished a domain join and is operating within the network boundaries of the domain. The member machine has network communication access to the domain controller.

4.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

UserClientMachineServiceTicket - The Kerberos service ticket for the member machine.

UserClientName – The account name provided by the user.

UserDomain– The domain name provided by the user.

UserDomainController – The domain controller (DC) information returned when locating a DC ([MS-DISO] section 5.3.4).

UserPassword– The password provided by the user.

UserTGT - The Kerberos ticket-granting ticket (TGT) for user logon.

UserGroupPolicy - Group policy to be applied to the user during logon.

UserDFSRef - DFS referral information for the user.

4.3.3 Task Abstract Parameters

This section describes data passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

LogonServer - The name of the domain controller to be used for authentication.

DNSServer - The name of the server to be used for DNS queries.

LDAPServer - The name of the server to be used for LDAP queries.

KDCServer - The name of the domain controller to be used for KDC queries.

ClientSite - The site within a domain where the client member machine resides.

SMBDialect - The SMB protocol dialect used for the secure channel between client member and DC.

4.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations

established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

UserLogonToken - The logon token created during user interactive logon, derived from UserClientMachineServiceTicket.

4.3.5 White-Box Relationships

The following figure represents the white-box relationships of the task within the client computer and with the domain controller.

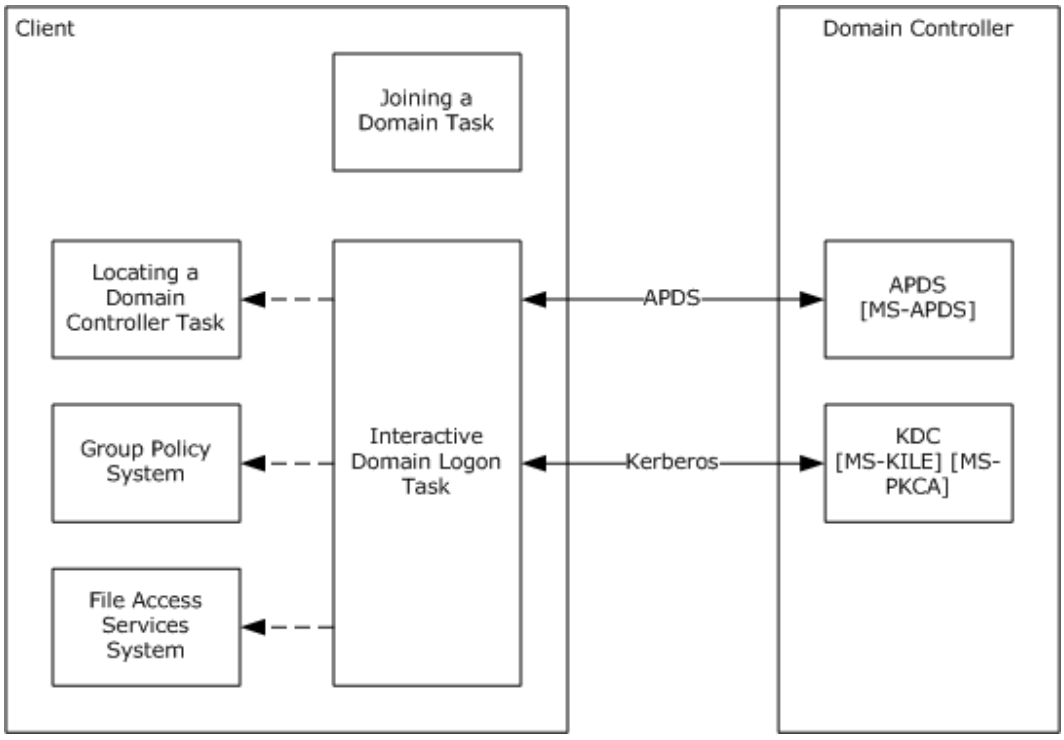

Figure 9: White-box relationships for the Interactive Domain Logon task

4.3.6 Task Events

This task initiates audit events based on the success or failure of the task. The audit events MAY be logged locally on the member machine and/or on the domain controller.

4.3.6.1 Task Timers

There are no timers employed directly by this task. Timers may be employed by invoked protocols (such as the protocol described in [MS-KILE]). Invoked protocol timers are detailed in their respective Protocol Technical Document.

4.3.6.2 Task Non-Timer Events

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

4.3.7 Task Architecture and Communication

The Interactive Domain Logon task uses a number of services that are used in the DomainJoin task described in [MS-DISO]. The following diagram shows communication between the local entity performing the task, the Client Computer, and the external entities providing services such as the DHCP server, DNS server, and domain controller.

The communication paths used during client computer system startup and domain join are shown for clarity and are part of the preconditions of this task.

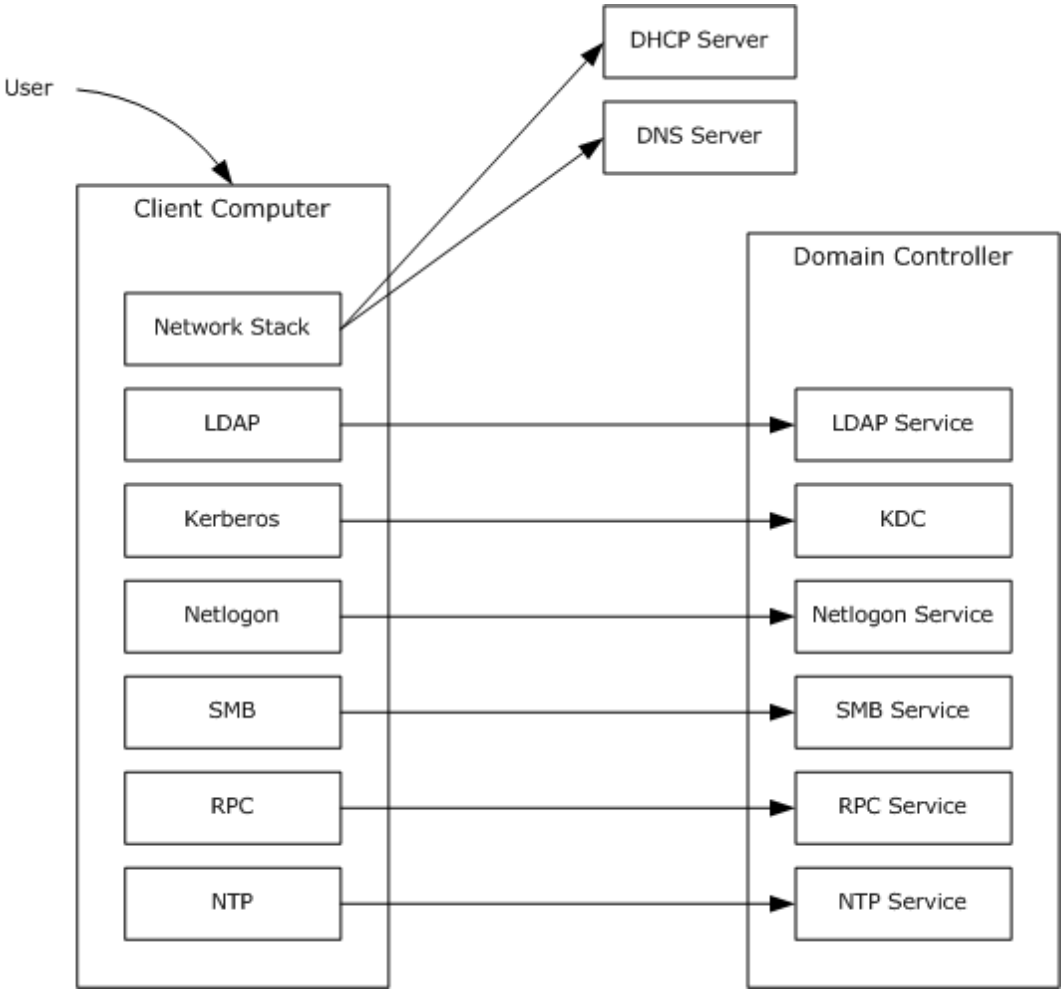

Figure 10: Communication paths for the Interactive Domain Logon task

4.3.8 Task Processing Rules

Init InteractiveDomainLogon:

Step 1: Check to see if client computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

Step 3: Client Computer displays UI for user logon and logon credentials.

Task InteractiveDomainLogon:

Step 1: User enters a set of domain credentials for client logon into the client logon process.

Step 2: Client logon process validates format of domain credentials for one of the acceptable formats.

Step 3: If the domain credential format is invalid, return error Bad_format.

Step 4: Client logon process uses the user-supplied domain credentials to create an authentication request to its domain controller.

Step 5: If an error is returned from Step 4, return error.

Step 6: Client logon process requests access to client computer resources and resources on the domain controller.

Step 7: Client logon process requests mapping/translation of the username.

Step 8: Client logon process requests policy information based on the mapped name from Step 7.

Step 9: Client logon process requests a list of the network mapped drives.

Step 10: Client logon process closes down connections to the DC that are no longer required.

Step 11: Client logon process creates an interactive processing session for the user.

Step 12: The user has now successfully interactively logged onto the client.

Each of the primary steps within this task can return errors. Appropriate error handling SHOULD be done.

4.3.9 Task Failure Scenarios

Each of the primary steps in the Interactive Domain Logon task can result in a failure case. Some of the errors are non-recoverable - for example, if invalid user credentials are supplied, the entire task is restarted.

Some of the protocols contain recoverable error processing that is handled in the lower level protocols. For example, using the protocol described in [MS-KILE], the KDC will be contacted during Step 4. The KILE protocol will retry contacting the KDC up to three times if there is no response. This failure/recovery path is handled at a lower protocol level and is not visible to the task.

4.4 Task Details

This section contains the details that complete the descriptions in earlier sections of the document. These details are needed to understand and implement this task. Section 4.4.4 contains a task component flow diagram to illustrate the step progression that is used in this task.

4.4.1 Task Precondition Details

This task has the precondition that the domain controller (DC) has been configured to support the domain infrastructure. The user account for the client computer interactive logon has been created and provisioned on the DC.

Also, as part of the Windows startup procedure for the client computer, the following actions have taken place (see [MS-DISO] for additional details):

- The client computer system has successfully connected to the network. At startup, the computer begins the process of connecting to the network and loads TCP/IP networking protocol stacks. TCP/IP can be configured to use either static or dynamic address configuration.
  - If dynamic TCP/IP configuration is used, the DHCP protocol will be used. DHCP is a well-documented protocol technology that is a core component of the Windows operating system.
  - If static TCP/IP configuration is used, then addressing information has manually been configured on the computer. Typically, static addresses are used for resources that do not change very often, such as routers and servers. Clients are often configured to use the DHCP protocol.
- The client computer has determined the IP address for DNSServer.
- Domain controller discovery and detection: A client computer needs to get the latest configuration status during each startup phase. There MUST be at least one DC in its domain to provide this information. In Windows domains, each controller is also an LDAP server. In order to retrieve a list of available controllers for the domain DnsDomainName, the client can query DNSServer for SRV resource records with the name _ldap._tcp.dc_msdcs.DnsDomainName. This SRV record is registered by the NetLogon service on the DC.
- Site determination: The client computer will attempt to find the closest site during the startup process and then acquires the name of the site. This name will be stored locally. If the client knows which site it belongs to, the client computer can send a query for DCs in its site to the DNS server. The client will use a series of LDAP queries to LDAPServer to determine its site.
- SMB dialect negotiation: The client will negotiate an SMB dialect with the DC. See [MS-SMB] sections 1.7 and 2.2.4.5.1 and [MS-SMB2] section 2.2.3.
- Secure Channel to the Domain Controller: When the client determines its site and DC, it can then create a secure channel with that DC. The secure channel is described in [MS-NRPC]. A secure channel is a connection between a domain member and a DC, LogonServer, that is established to retrieve domain-specific information, to update computer-specific information in Active Directory, such as computer passwords, and to validate domain membership. The protocol described in [MS-NRPC] is an RPC interface and as a result, has a prerequisite that [MS-RPCE] specifies. To use the protocol described in [MS-NRPC], the client possesses a shared secret with the DC.
  - The client proceeds by connecting to the Netlogon interface of the target by using Session Key Negotiation ([MS-NRPC] section 3.1.4.1). The End Port Mapper is involved in this process in order to connect the client with the correct port on the controller. Finally, the client sends three calls to the interface: NetrServerReqChallenge, NetrServerAuthenticate3, and NetrLogonGetdomainInfo.
- Kerberos Tickets: After the secure channel has been established, the client will retrieve all necessary tickets from the DC. Because all Windows DCs are a Kerberos Key Distribution Center (KDC), the client detects the KDC, KDCServer, in a similar fashion as the LDAP server, LDAPServer. To acquire the KDC, an LDAP query of the form _kerberos._tcp.Default-First-Site-Name._sites.dc._msdcs.DnsDomainName, where Default-First-Site_Name is the site name for the

computer, is made. The client performs an Authentication Service AS request to KDCServer, followed by a ticket request for the DC and the Kerberos service running on the controller (krbtgt).

- IPC$: The client connects to IPC$ share on the DC.
- Distributed File System (DFS): The client makes a DFS referral. A DFS referral is the protocol described in [MS-DFSC] that the client uses to get DFS-specific information that exists on the DC.
- The client performs a name translation and obtains its **Distinguished Name (DN)**.
- Group Policy: The client performs an LDAP request on the DC for RootDSE and loads Group Policy information for this particular client using the SMB protocol.
- PKI configuration: The client determines PKI configuration and autoenrollment. The status of the client's certificates is verified, and enterprise Certificate Authority (CA) certificates are downloaded.
- Time synchronization: The client sets the clock with the NTP protocol.
- DNS name update: The client obtains the DNS server that has authority for the client's zone and performs a dynamic DNS update based on [RFC2136].
- Connection Cleanup: The client closes open connections to the DC.
- The computer has successfully joined a domain as described in [MS-DISO].

4.4.2 Task Initialization of External Entities

This task does not initialize external entities. The protocols used by this task have been initialized during system startup and during the domain join task.

4.4.3 Task Event Details

4.4.3.1 Task Timer Details

There are no timers employed directly by this task. Timers may be employed by invoked protocols (such as that described in [MS-KILE]). Invoked protocol timers are detailed in their respective Protocol Technical Document.

4.4.3.2 Task Non-Timer Event Details

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has been completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

4.4.4 Task Architectural Details

The following figure details the steps that make up the task in a flow diagram. The protocols have been initialized during client computer system startup and during domain join.

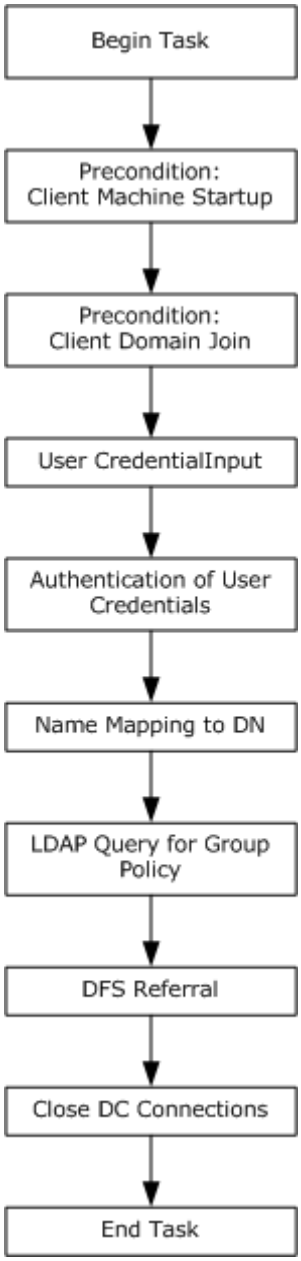

Figure 11: Flow diagram for the Interactive Domain Logon task

Interactive logon can be performed in a number of ways: through the Netlogon RPC interface [MS-NRPC], with password-based authentication; through Kerberos [MS-KILE] [RFC4120], with passwords; or through Kerberos PKINIT [MS-KILE] [RFC4556], using an X.509 public key credential. This example shows the password-based Kerberos exchanges.

The general flow of the interactive logon is shown in the following example and diagram.

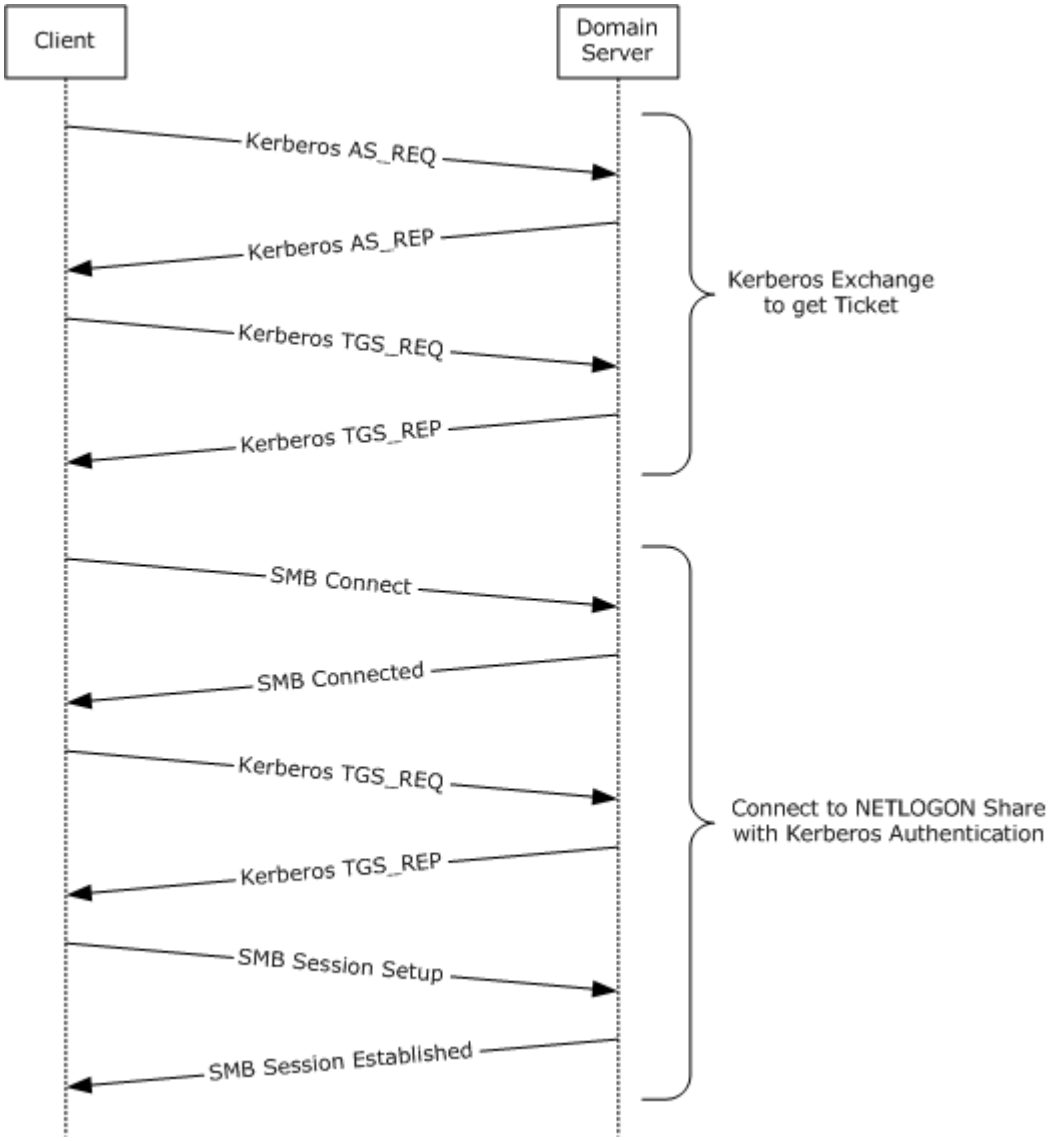

Figure 12: Sequence diagram for the Interactive Domain Logon task

The user enters credentials on the domain client machine: the account name (UserClientName), domain name (UserDomain), and password (UserPassword). The domain client then locates a domain controller and requests an initial ticket from the Kerberos KDC service on the domain controller. This exchange produces a ticket-granting ticket (TGT), which is encrypted such that only the domain controller can decrypt it; the domain client cannot yet determine that the user has entered valid credentials. Therefore, the domain client invokes Kerberos again, using the TGT, and requests a ticket for itself, using the TGT for the user. The domain client can decrypt that ticket, and after doing that can validate that the user did provide correct credentials.

Once a valid TGT is available, the domain client can then use the credentials for authenticating additional traffic. For example, the user might be administratively configured to have a script or program run immediately after logon. The domain client would then make a CIFS/SMB connection to the domain controller server's NETLOGON share, and attempt to execute the logon script.

4.4.4.1 Location

The *UserDomain* element is used to seed the locator algorithm and the desire to find a domain controller with a Kerberos KDC on it. Upon completion, the *UserDomainController* element is populated with the domain controller address, or an error has occurred and the logon attempt MUST fail.

If the *UserDomain* and *Domain* name ([MS-DISO] section 4.3.1.1) elements are the same, then the same domain controller values can be used for both the domain client computer and the user. If not, then the locator is seeded again with the *UserDomain* element and the location process executes again. Upon completion, the *UserDomainController* element is populated with the results, or an error has occurred and the logon attempt MUST fail.

4.4.4.2 Kerberos Exchange

The logon attempt is made through the Kerberos protocol. The Kerberos protocol is initialized with the Client Name (*cname*) set to *UserClientName*, and the Client Realm (*crealm*) is set to the **fully qualified domain name (FQDN)** of *UserDomain*. The password is taken from *UserPassword*. The KDC to use is from *UserDomainController.Address*. The Kerberos AS subprotocol ([RFC4120] section 3.1) is executed, resulting in a TGT for the user. The TGT has the PAC with information about the user, but that is not accessible yet because the TGT is encrypted such that only the KDC can decrypt it.

The KDC for the domain to which the user belongs constructs the TGT according to [RFC4120] section 3.1, but also such that it contains a PAC as specified in [MS-PAC]. The user account may be subject to administrative policy such as valid times of day to log on; the KDC implementation SHOULD honor those policies if implemented. The PAC will be filled out completely, subject to the policy of the domain itself. For example, if the user has a valid logon script administratively assigned, this value will show up in the PAC. If there is no script, then the field will be left empty as specified in [MS-PAC].

The next step is for the domain client to request a ticket to the workstation itself. In this way, the resulting ticket can be decrypted by the domain client to validate the user. The ticket request is constructed as follows: the Kerberos ticket-granting service (TGS) subprotocol ([RFC4120] section 3.3) is invoked, with a server name of "host/" and the hostname of the client, in this example, "host/workstation". The server realm is *DomainName.FQDN* from the domain client ADM, in this example, "corp.example.com".

The entire TGS subprotocol is executed, which includes the possibility of traversing cross-realm trusts. The TGS subprotocol completes with a service ticket for "host/workstation.corp.example.com", or encounters an error. Any error not recoverable within the TGS protocol is a fatal error, and the logon process MUST fail.

The domain client can decode the service ticket based on the domain client's *Password* element from the client ADM ([MS-DISO] section 4.3.1.1). When decoded, the PAC [MS-PAC] can be recovered from the ticket and interpreted on the domain client.

4.4.4.3 Connection to the NETLOGON share

Connection to the NETLOGON share is triggered by the existence of a logon script. In this example, the user has been configured with a logon script called "domain-logon.bat". The domain client determines that a logon script has been configured by examining the *LogonScript* field of the KERB_VALIDATION_INFO structure as specified in [MS-PAC] section 2.5. The *LogonScript* value is the name of a script to execute, relative to the NETLOGON share on the domain controller.

The domain client establishes a CIFS/SMB connection to the domain controller of the user. This is done by seeding the connection information to the CIFS/SMB client on the domain client, specifying *UserDomainController.Name* as the target server, and "NETLOGON" as the share on the server. The normal CIFS/SMB session establishment outlined above takes place. The connection is performed in the context of the newly logged-on user, so the credentials that are used are those that the TGT just retrieved in the previous section.

4.4.5 Task Processing Rule Details

Once the client's system startup and domain join have been completed as summarized in section 4.4.1, the user can initiate the Interactive Domain Logon task. The overall sequence is as follows:

Step 1: The user enters a valid set of domain credentials for client logon. These credentials will be one of the following:

- Domain, username, and password
- Smart card certificate and PIN

In the case of password-based logon:

- Step 2_PW: The client logon process validates the name format of domain credentials for one of the acceptable formats. Valid formats include SAMAccountName and Domain or UPN (implicit or explicit formats).
- Step 3_PW: If the domain credential format is invalid, return error Bad_format.

In the case of smart card-based logon:

- Step 2_SC: The smart card subsystem unlocks the smart card with the PIN.
- Step 3_SC: If the PIN is invalid, return error Bad_PIN.

Step 4: The domain member machine uses the user logon credentials to generate a Kerberos authentication request, KRB_AS_REQ [MS-KILE] [MS-PKCA]. If the KRB_AS_REQ fails, then the machine can use NTLM Interactive Logon ([MS-APDS] section 3.1.5.1).

In the case of Kerberos authentication:

- Step 4_Kerb: The KDC will process the KRB_AS_REQ and validate the user credentials and check for account restrictions, such as the account being disabled, logon hours restriction, or the account being locked out.
- Step 4_Kerb: If successful, a KRB_AS_REP will contain a TGT for the user.
- Step 4_Kerb: The domain member machine logon process requests the Kerberos ticket (KRB_TGS_REQ) for the member machine. The member machine ticket is passed to the client security component in the operating system (for Windows that is the LSA) for use in creating a logon token on the machine, **UserLogonToken**. The specific service ticket requested are for:

```
Kerberos:Server Name = <clientname> $
```

In the case of NTLM Interactive Logon:

- Step 4_NTLM: The DC will process the NETLOGON_INTERACTIVE_INFO message and validate the user credentials and check for account restrictions, such as the account being disabled, logon hours restriction, or the account being locked out.
- Step 4_NTLM: If successful, it will return a NETLOGON_VALIDATION_SAM_INFO4 or NETLOGON_VALIDATION_SAM_INFO2 or NETLOGON_VALIDATION_SAM_INFO.

Step 5: If error is returned from Step 4, return the error.

Step 6: The domain member machine uses IDL_DRSCrackNames [MS-DRSR] to perform a name translation from the username provided in Step 1 into a DN.

Step 7: Group Policy information is retrieved via the LDAP and SMB protocols. An LDAP query with the DN from Step 6 is made against the LDAPServer to obtain information on the group to load. The client logon process establishes an SMB connection to the controller and downloads the necessary group policy information for the user, UserGroupPolicy.

Step 8: DFS referral information, UserDFSRef, will be retrieved via [MS-DFSC].

Step 9: If logon scripts exist, script is retrieved (section 4.4.4.3) from the domain controller.

Step 10: The SMB connections to the domain controller are closed.

Step 11: The client logon process creates an interactive processing session for the user.

Step 12: The user has now successfully logged on to the client.

Each of the primary steps within this task can return errors. Appropriate error handling SHOULD be done.

4.5 Task Security

This section documents security issues specific to this task that are not otherwise described in the Technical Documents (TDs) for the protocols used in the task. It does not duplicate what is already in the protocol TDs unless there is some unique aspect that applies to the system as a whole.

The logon token, UserLogonToken, that is created within the LSA with the user's service ticket for the client member machine can be used for any process that is associated with the interactive logon session. Care is to be taken to protect this logon token from other processes that are not associated with the interactive logon session and to prevent tampering with the logon token by any process.

5 HTTP Access Authentication - Server Task

This section describes the HTTP Access Authentication - Server task. HTTP 1.1 [RFC2616] authentication is addressed in [RFC2617]. HTTP is an RFC standard used internationally for Internet web servicing. The general topology of the HTTP protocol is that of a client/server role. The **HTTP client** makes requests that are sent to the **HTTP server**. The HTTP server can enforce authentication requirements on the HTTP requests. If a request lacks valid authentication material (specifically, in the HTTP header), the HTTP server generates a Challenge message (token), which is sent to the HTTP client. The HTTP client can then form a ChallengeResponse token based on user credentials applied to the authentication protocol. The initial Request along with the ChallengeResponse token is sent again to the HTTP server. The HTTP server can then validate the ChallengeResponse token in processing the Request. This task will focus on the HTTP server side of the authentication.

HTTP authentication [RFC2617] contains specification details on two forms of authentication, Basic and Digest Authentication. The HTTP authentication framework is extensible to other authentication mechanisms. This provides the opportunity to extend the types of authentication available for use in HTTP requests.

Authentication is also possible via Forms Based Authentication. Here the user credentials (username and password) are entered by the user in an HTTP format and transmitted in cleartext to the HTTP server, usually via a secure HTTPS connection. The HTTP server can then validate the user credentials. This type of authentication is not part of the HTTP authentication protocol [RFC2617] and will not be covered in additional detail.

5.1 Task Overview

5.1.1 Task Purpose

This task provides the steps that the server undertakes for HTTP Web Access Authentication. To enforce access controls over files and resources on an HTTP server, the server will recognize the validated identity of the requestor, and the files and resources can be configured for access control and enforced authentication and authorization.

5.1.2 Task Applicability

This task is applicable to providing access controls over server system resources that are made available via the HTTP protocol.

5.1.3 Task Use Cases

5.1.3.1 Stakeholders and Interests Summary

The stakeholders for this task are:

Server computer/ Server - The server is any server other than a domain controller that is a member of the domain and offering services to clients in the domain. The domain controller can be the server, but for protocol evaluation, the domain controller and the server will be considered separate.

Client - As described in section 4.1.3.1.

5.1.3.2 Supporting Actors and Task Interests Summary

The supporting actors for this task are:

Domain Administrator - Same as described in section 4.1.3.2. For this task, this entity will provision the domain credentials for the user to accomplish the task of access to network resources.

Server Administrator - The server administrator is responsible for the server itself. The server administrator manages the server and determines the resources that are managed by the server and the associated policies for those resources. The server administrator relies upon the domain (and domain controller) to publish necessary rendezvous information for clients to find the server, to authenticate the clients, and to provide the necessary authorization information for the server to manage its resources. Depending on the server, publication of information may or may not be required. For this task, this entity will provision the HTTP server to require authentication for local resources and for them to be made available to the HTTP service.

User - This entity is described in section 4.1.3.1. For this task, this user initiates the task action on the client. The task is being performed at the request of the user to gain access to the resource server, HTTP server, for information on behalf of the user.

Domain Controller - As described in section 4.1.3.2. For this task, this domain computer provides validation of the user authentication, and upon successful validation, also provides user-specific authorization information.

5.1.3.3 Use Case Diagrams

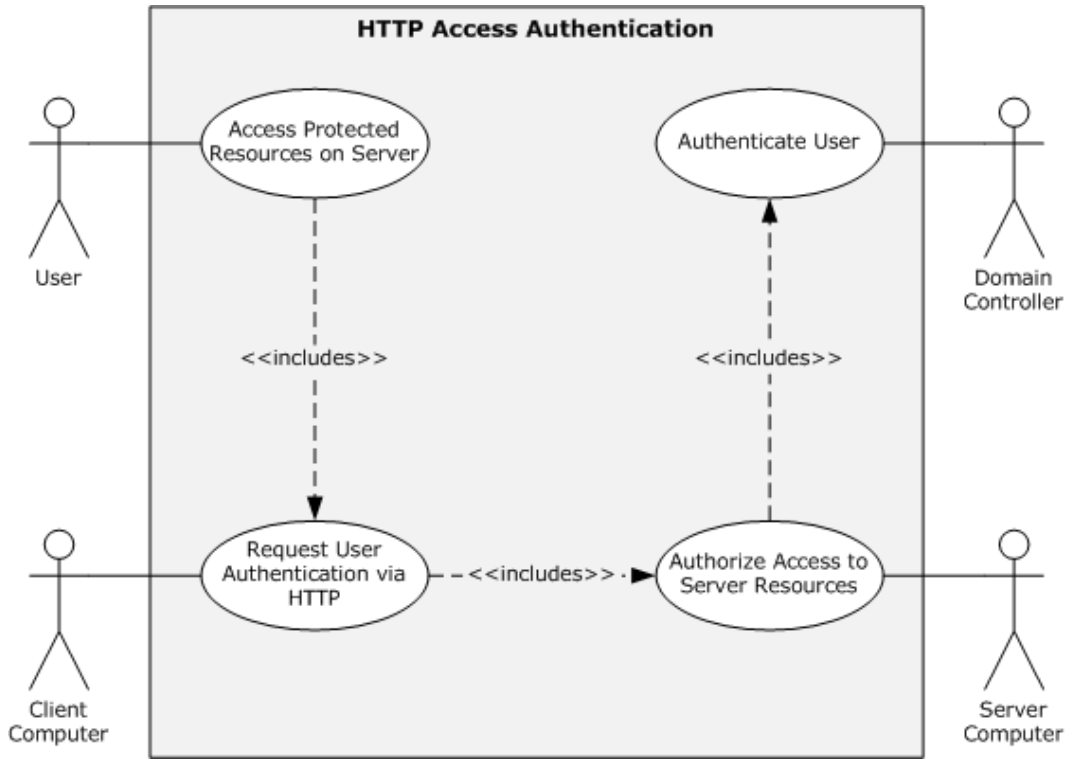

Figure 13: HTTP Access Authentication - Server use case diagram

Here the user is initiating an HTTP request from the HTTP client running on the client computer. The client computer may be joined to the domain administered by the domain controller. The server computer is running the HTTP server service. The administration of the HTTP server may be done locally by the server administrator or by policy by the domain administrator through various domain services.

The server administrator will set up and provision the HTTP server on the server computer. The administrator will determine which resources (files) would need authorization to access and set the appropriate HTTP server configuration options to require authentication. The administrator can also configure which forms of authentication are acceptable.

The domain administrator needs to provision the user accounts. Part of this provisioning is to grant the user specific rights appropriate for the user. These rights will be sent over to the HTTP server on successful authentication and evaluated against the requirements placed on the server resources.

5.1.3.4 HTTP Access Authentication - Server

For the Use Case of HTTP Access Authentication - Server, the user on the client is initiating the action with the goal of access to protected resources on an HTTP server. The server will be configured for acceptable authentication mechanisms. The arrival of the HTTP request will initiate the task on the server. The HTTP server will deny unauthorized access to protected resources and respond back to the HTTP client with an authentication challenge. The user will provide credential material to respond to the authentication challenge. The server receives the HTTP request with authentication material from the authentication protocol on the client computer, which can then be evaluated. If authentication is successful, the task is complete. Subsequently, the system can then evaluate if the user is authorized to access the requested resources. If the user is authorized to access, the requested resources are communicated back to the HTTP client.

Goal: The user needs authentication to access protected resources on an HTTP server.

Context of Use: This is typically done when the user opens a webpage.

Direct Actor: The user is the Direct Actor.

Primary Actor: Same as the Direct Actor.

Supporting Actors: The Domain Administrator, Server Administrator, and Domain Controller are the Supporting Actors.

Stakeholders and Interests:

- The User who is attempting to access protected HTTP resources.
- The Domain Controller that is available to authenticate the user.

Precondition:

- The domain controller is available.
- The user account is provisioned.

See section 5.2.3 for detailed information.

Minimal Guarantees: The user will be authenticated, or the authentication will fail.

Success Guarantee: The user is authenticated.

Trigger: The user attempts to open a webpage.

Main Success Scenario:

1. The user opens a webpage.
2. The browser sends an HTTP request to the HTTP service.

3. The server sends an authentication request to the domain controller.

4. The domain controller authenticates the user.

Extensions: None.

5.2 Task Context

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

HTTP Access Authentication task is a required action for successful access to protected resources via the HTTP protocol.

5.2.1 Task Environment

HTTP provides an extensible mechanism for HTTP level authentication. Several authentication protocols are offered as extensions to the Basic and Digest Auth described in [RFC2617]. These protocols are encapsulated within the HTTP headers of the HTTP request and response messages.

5.2.2 Task Relationships

5.2.2.1 Black Box Relationship Diagrams

The HTTP Access Authentication - Server task uses a number of protocols (some are prerequisite and others are used within the task itself). The HTTP server initiates the task based on the arrival of the HTTP request from the HTTP client. The task involves a server member machine in the domain as well as the authority for the domain, that is, the domain controller. The HTTP client may be either a domain-joined computer or a non-domain-joined computer.

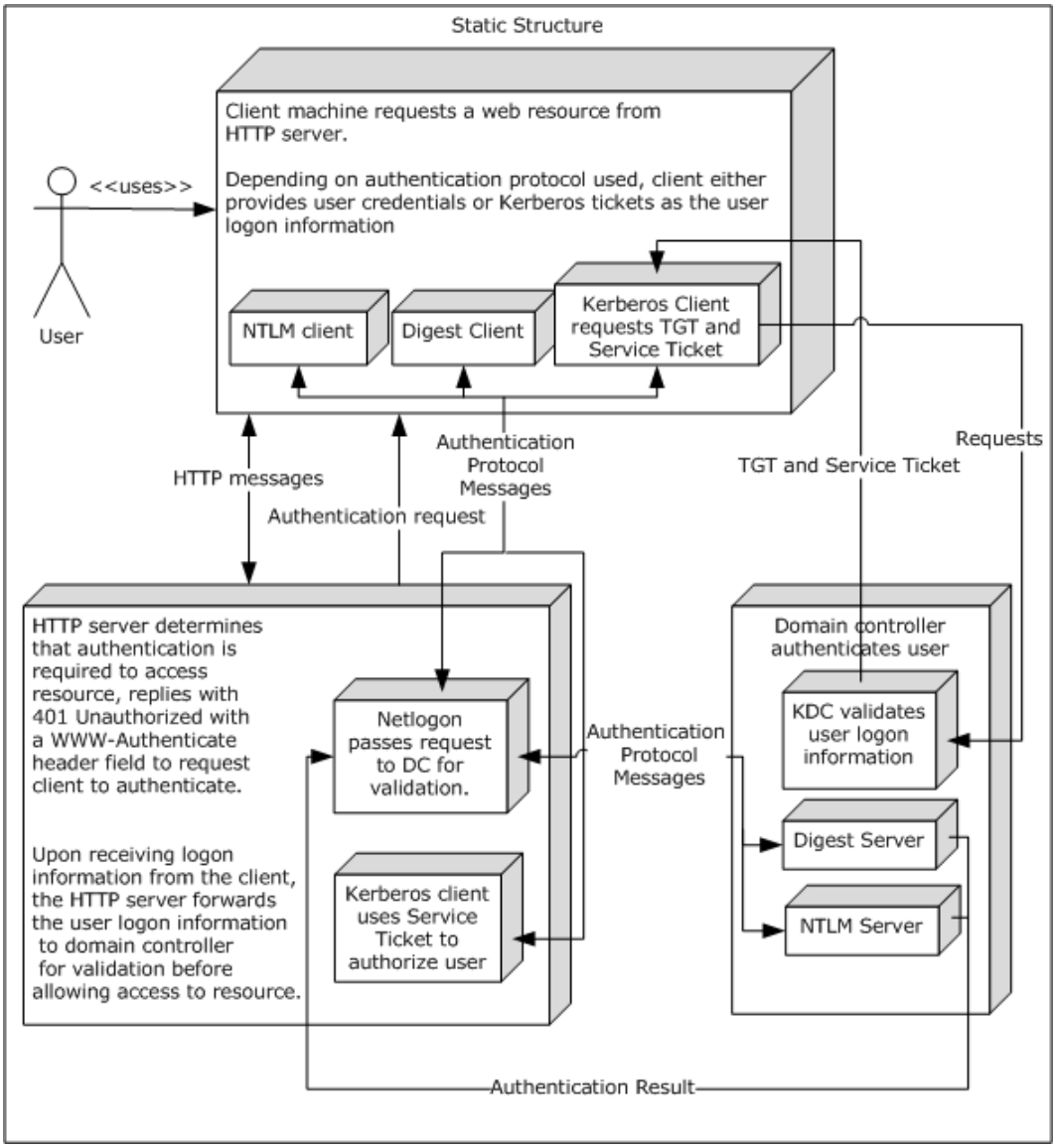

Figure 14: Black box relationships for the HTTP Access Authentication - Server task

5.2.2.2 Task Dependencies

This task has a number of external dependencies. The following protocols are used by this task:

- Precondition - System Level
  - Domain Interaction System Overview [MS-DISO]
- Precondition - Protocols
  - TCP/IP
  - DNS

- HTTP
- Digest Authentication
- Negotiate
- NTLM
- Kerberos
- Group Policy

- Precondition - Tasks
  - Server Domain Join
- External Server System
  - Active Directory
  - DNS Directory
  - KDC

5.2.2.3 Task Influences

The HTTP Access Authentication - Server task has several external influences. These influences can alter the state of the protocols, for example limiting which authentication mechanisms are offered.

External influences are as follows:

- Group Policy for the computer
- Group Policy for the user account for the domain logon
- Active Directory - account lockout subsystem, account restrictions
- Security event logging on the domain controller
- HTTP resource access requirements

5.2.3 Task Assumptions and Preconditions

This task has the precondition that the domain controller has been configured to support the domain infrastructure. The user account for the client authentication has been created and provisioned on the domain controller.

Also, as part of the Windows startup procedure for the server computer, the following actions have taken place (see [MS-DISO] for additional details):

- Successful connection to the network and startup of networking stacks.
- Successful domain join.
- Establishment of a secure channel with the domain controller.

Additionally, the HTTP server has configured and started up the HTTP service, which will respond to HTTP requests from the client.

The client computer may be either domain-joined or not joined to a domain. The client computer will have access to the KDC if the KILE protocol is to be used.

5.2.4 Task Versioning and Capability Negotiation

HTTP requests can have multiple authentication challenges in the HTTP headers. The HTTP server is configured with the authentication mechanisms to include in the HTTP headers of the challenge.

The negotiation protocol [MS-SPNG] selects between several underlying protocols - specifically Kerberos and NTLM.

Underlying protocols MAY perform versioning as part of the authentication. The details would be contained in the respective protocol TDs.

5.3 Task Architecture

The basic architecture contains a domain controller, a member server computer, and a client computer. The HTTP server is initiating an HTTP Access Authentication - Server task to validate the user credentials and obtain authorization information.

5.3.1 Task Architectural Constraints

The primary constraint is that the computer has successfully accomplished a domain join and is operating within the network boundaries of the domain. The member machine MUST have network communication access to the domain controller.

5.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by the processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

RequestedResource - Requested resource target name extracted from the HTTP request.

ClientAuthProtocol - Authentication mechanism used by HTTP.

ClientAuthMechanism – extracted from auth-scheme in the HTTP request.

ClientAuthToken – extracted from auth-param in the HTTP request.

ServerSecurityContext - A security context that represents the server's authentication security state.

ServerAuthenticationMechanism - A list of authentication mechanisms that may be used for HTTP client authentication. The Windows operating system supports Basic, Digest Auth, Negotiate (for which Kerberos may be offered) and NTLM.<1>

5.3.3 Task Abstract Parameters

This section describes data that is passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

LogonServer - The domain controller to be used for authentication.

DNSServer - The domain controller to be used for DNS queries.

RequestedResource - The target resource specified by the client in the HTTP request.

5.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

UserLogonToken - A logon token created during user network logon, derived from the authentication protocol and validated account information from the domain controller.

5.3.5 White-Box Relationships

The following figure represents the white-box relationships of the task within the client computer, server computer and with the domain controller.

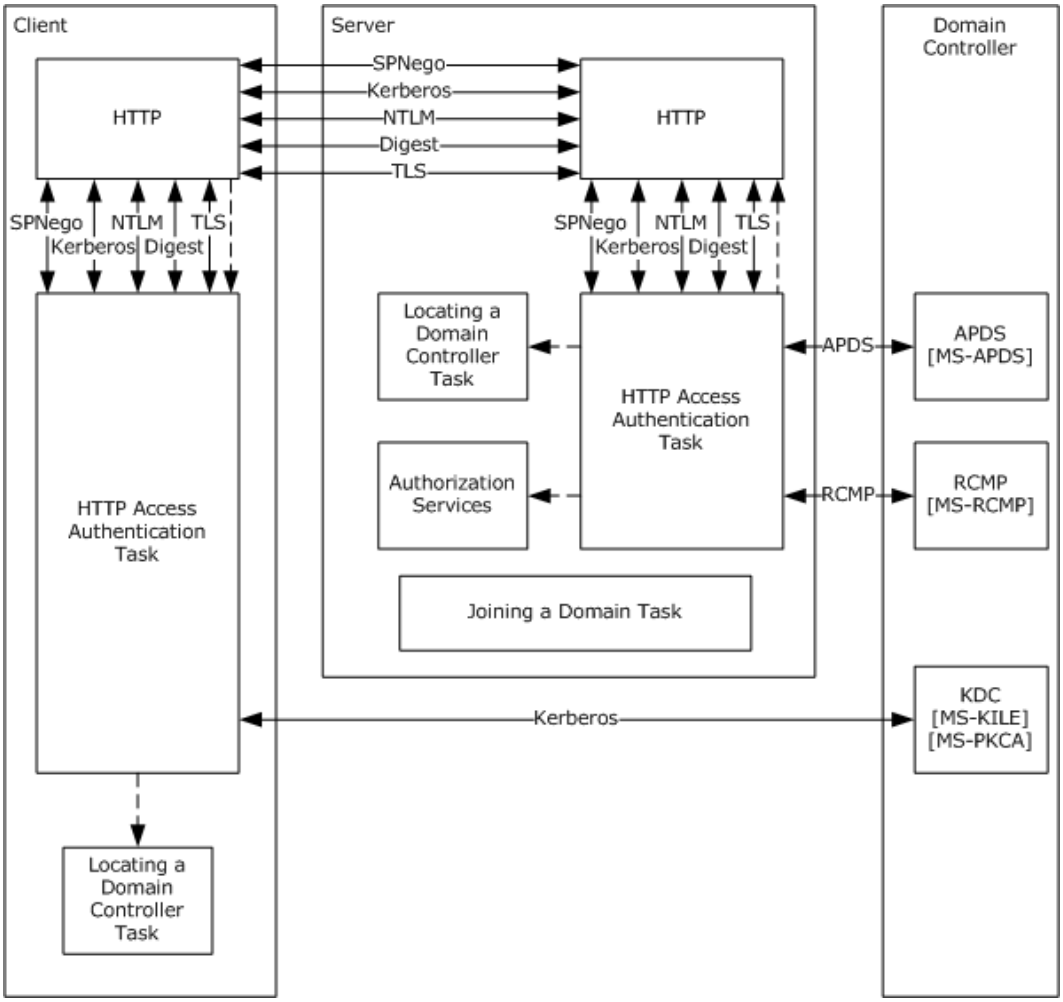

Figure 15: White-box relationships for the HTTP Access Authentication - Server task

5.3.6 Task Events

This task initiates audit events based on the success/failure of the task. The authentication audit events are logged at the domain controller for the verification of the user credentials. Authentication audit events and access check events are also logged at the server computer.

5.3.6.1 Task Timers

There are no timers employed directly by this task. Timers may be employed by invoked protocols (such as KILE). Invoked protocol timers are detailed in their respective Protocol Technical Document.

5.3.6.2 Task Non-Timer Events

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

5.3.7 Task Architecture and Communication

The HTTP Access Authentication - Server task uses a number of services that are also part of the DomainJoin task described in [MS-DISO]. The following diagram shows communication between the local entity performing the task, the client computer, and the external entities providing services such as the DHCP Server, DNS Server, and Domain Controller.

The communication paths used during client computer system startup and domain join are shown for clarity and are part of the preconditions of this task.

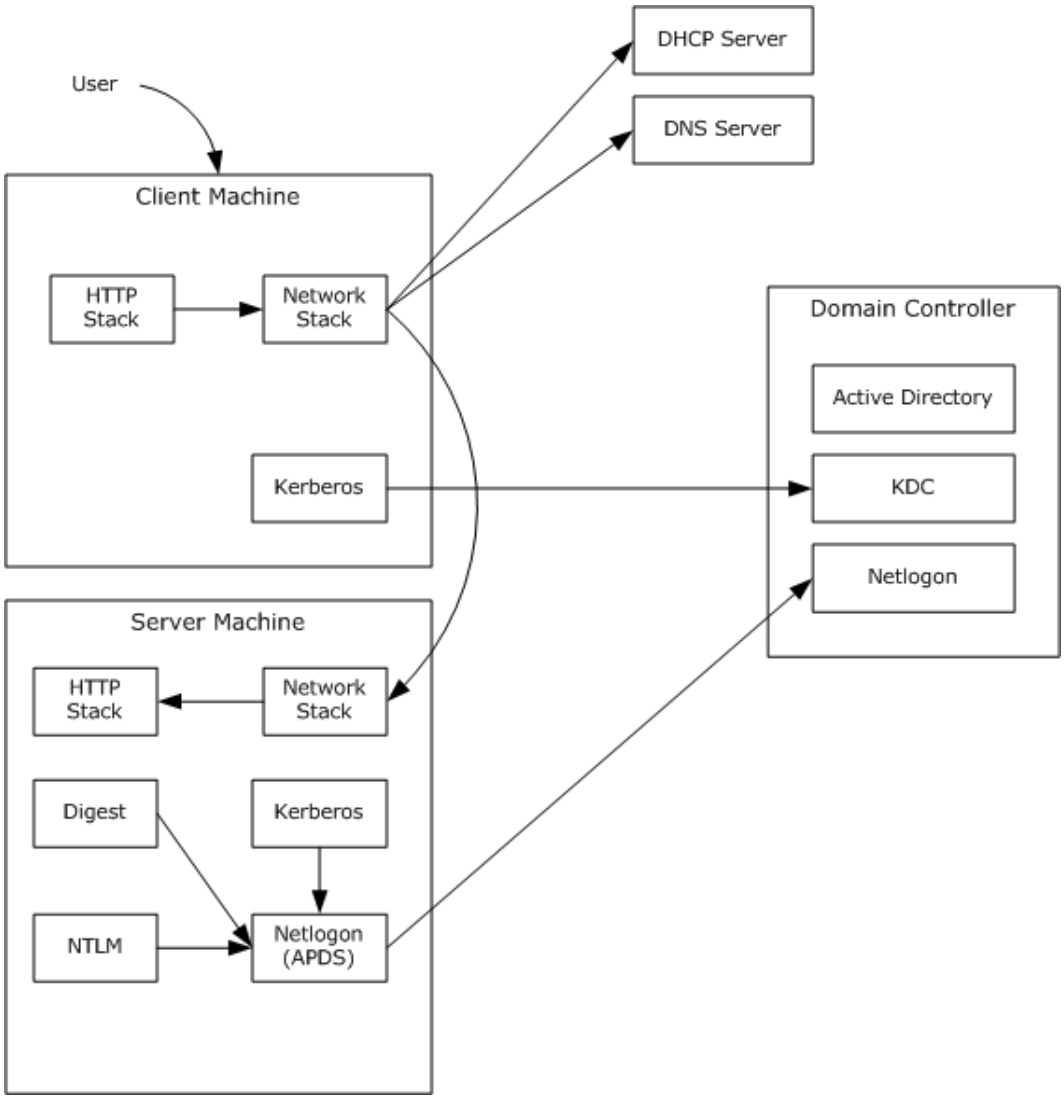

Figure 16: Communication paths for the HTTP Access Authentication - Server task

5.3.8 Task Processing Rules

Init HTTPAccessAuth:

Step 1: Check to see if server computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

Step 3: HTTP server computer starts up HTTP service.

Step 4: Client computer sends an HTTP request to the server.

Step 5: Server receives HTTP request from the client.

Task HTTPAccessAuth:

Step 1: Server extracts requested resource target name from the HTTP request.

Step 2: Server determines if authenticated access is required for the requested resource.

Step 3: If no authentication is required for access, return Success and exit HTTPAccessAuth task.

Step 4: Service extracts the authentication information attached to the HTTP request.

Step 5: If there is no authentication information, go to Step 12.

Step 6: Service determines which authentication mechanism is being used in authentication information.

Step 7: Service sends the authentication information to the specified authentication protocol.

Step 8: If the authentication protocol returns an error, go to Step 12.

Step 9: If the authentication protocol returns status that additional data is required, HTTP response will be sent to the client and the task will be exited.

Step 10: Service verifies authenticated response with access check for resource.

Step 11: If Access check is successful, return success and exit the task

Step 12: HTTP server generates an authentication challenge to send back to HTTP client and returns error of Access_Denied.

Each of the primary steps within this task can return errors. Appropriate error handling SHOULD be done.

5.3.9 Task Failure Scenarios

Each of the primary steps in the HTTP Access Authentication - Server task can result in a failure case. On any error not directly specified in the processing tasks, the task MUST return System_Error error and fail the HTTP request.

Some of the protocols contain recoverable errors that are handled in the lower-level protocols. For example, in the KILE protocol, the KDC will be contacted during Step 7. The KILE protocol will retry contacting the KDC up to three times if there is no response. This failure/recovery path is handled at a lower protocol level and is not directly influenced by the task.

5.4 Task Details

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

5.4.1 Task Precondition Details

This task has the precondition that the domain controller has been configured to support the domain infrastructure. The user account for the client member logon has been created and provisioned on the domain controller.

Also, as part of the Windows startup procedure for the server computer, the actions specified in section 4.4.1 have taken place (see [MS-DISO] for additional details). The computer has successfully joined a domain.

Step 1: Check to see if the server computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

The following additional steps will be undertaken:

Step 3: The server computer starts up the HTTP service. The server is required to run a service to support HTTP **protocol server**-side operations. The service MUST implement [RFC2616] and [RFC2617].

Step 4: The client computer sends an HTTP request to the server. The client user initiates an HTTP request conforming to [RFC2616] and [RFC2617].

Step 5: The server receives the HTTP request from the client.

5.4.2 Task Initialization of External Entities

This task does not initialize external entities. The protocols used by this task have been initialized during system startup and during the domain join task.

5.4.3 Task Event Details

5.4.3.1 Task Timer Details

There are no timers employed directly by this task. Timers may be employed by invoked protocols (such as KILE). Invoked protocol timers are detailed in their respective Protocol Technical Document.

5.4.3.2 Task Non-Timer Event Details

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

5.4.4 Task Architectural Details

The following figure details the steps in a flow diagram that make up the task.

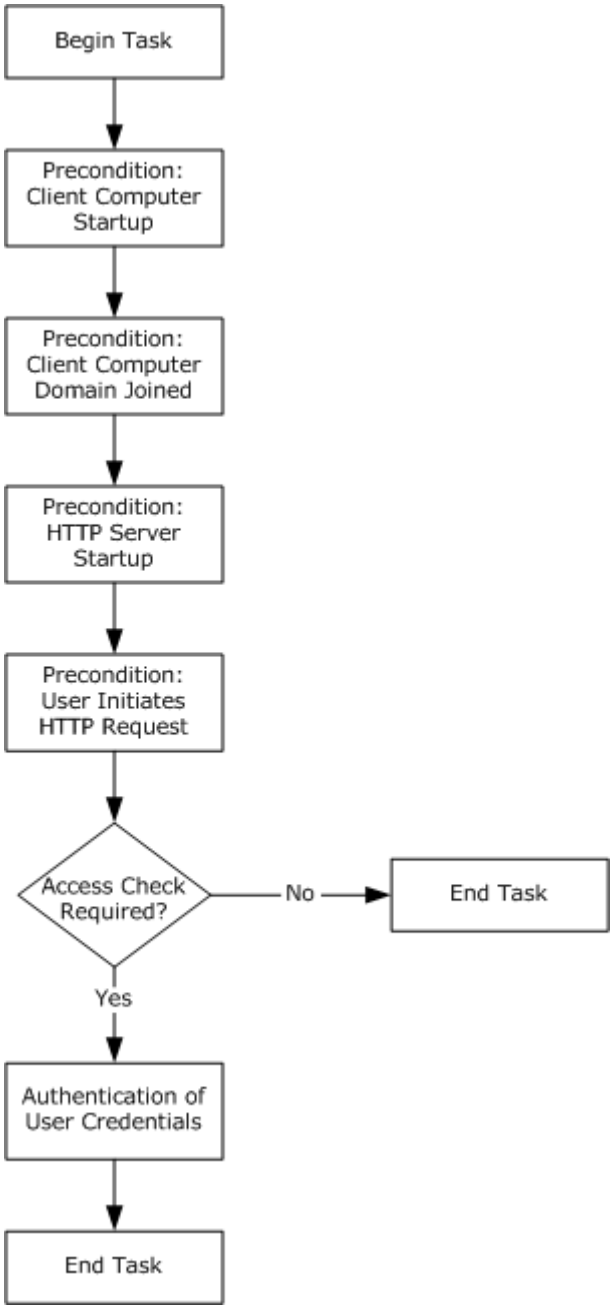

Figure 17: Flow diagram for the HTTP Access Authentication - Server task

The following figure illustrates the protocols that can be used in this task. The security protocols have been initialized during server computer system startup and during domain join; specifically, Kerberos has detected the KDC for the domain.

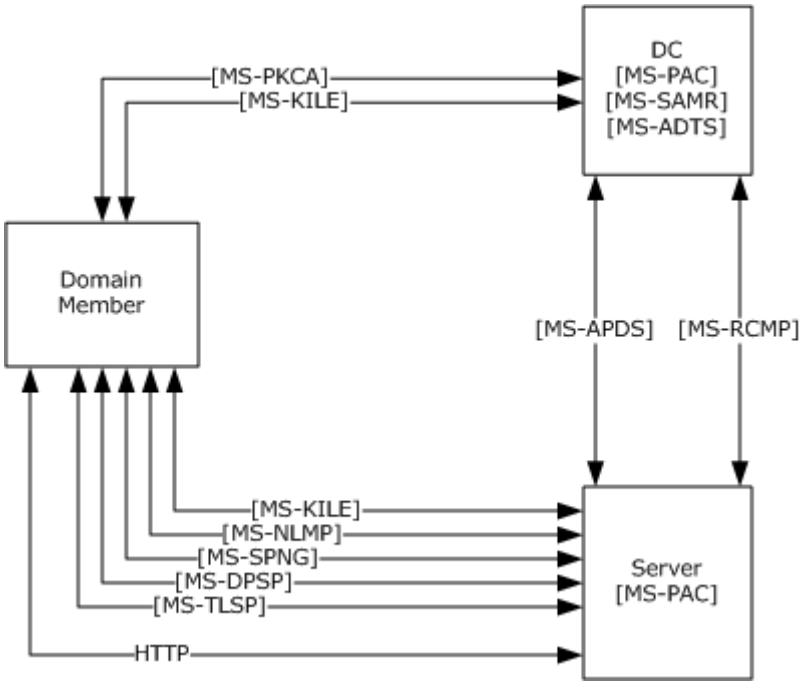

Figure 18: Standards used by the HTTP Access Authentication - Server task

The server may offer one or more mechanisms from the list of supported authentication protocols, ServerAuthenticationMechanism. These mechanisms are offered according to [RFC2617], [RFC4559], and [MS-N2HT]. The client MAY also include authentication information into the HTTP request via these same standards and TDs.

The authentication token information is contained within the HTTP request and response according to these standards and TDs.

The server, depending on the authentication mechanism used in the HTTP request, will use [MS-APDS] section 1.4, as described in the document.

The following diagram details the protocol traffic between the HTTP client, HTTP server, and domain controller when the HTTP client is using NTLM authentication.

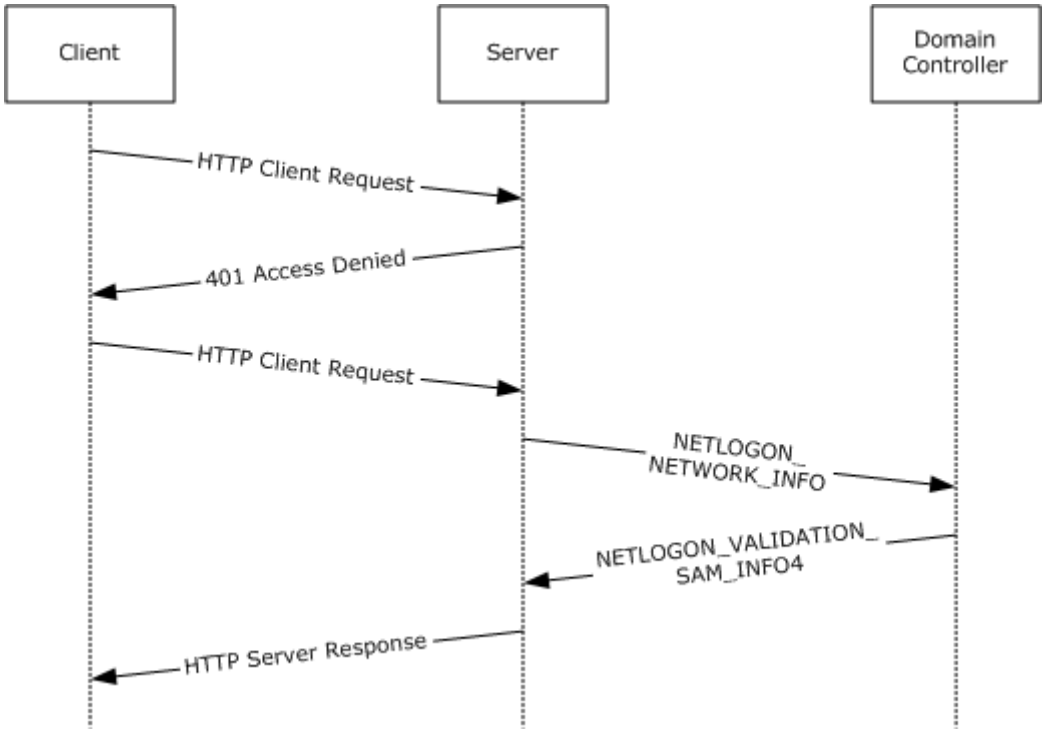

Figure 19: Protocol traffic for NTLM authentication

As a second example, the protocol traffic between the HTTP client, HTTP server, and domain controller when the HTTP client is using Digest authentication can be illustrated as follows.

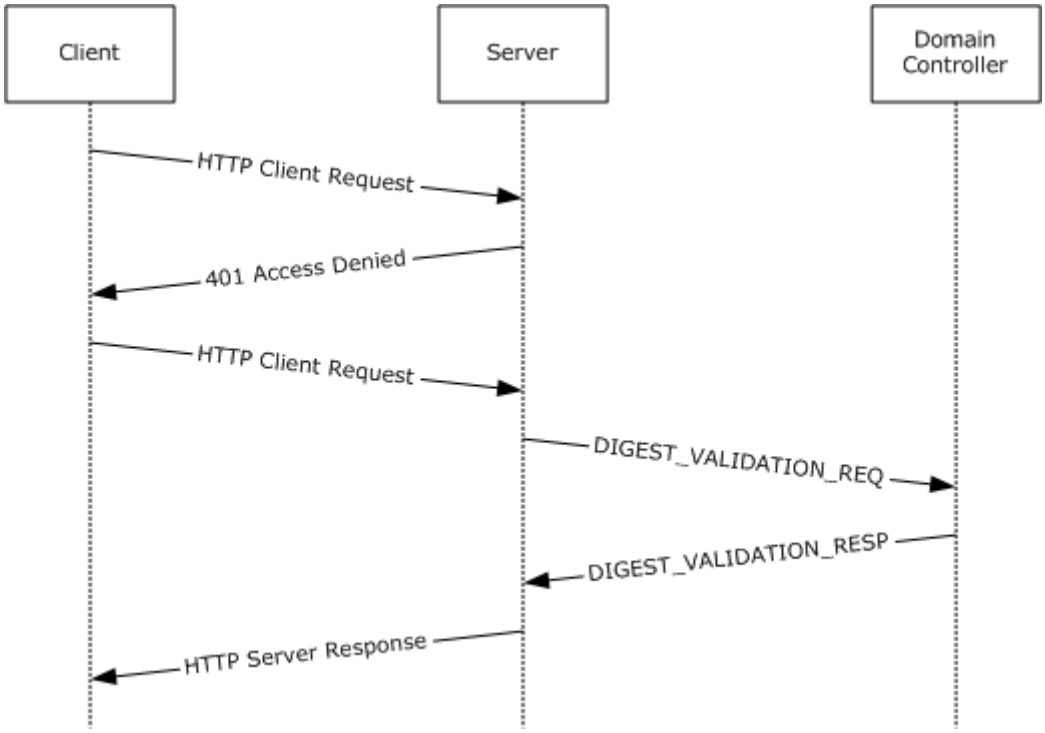

Figure 20: Protocol traffic for Digest authentication

As a third example, the protocol traffic between the HTTP client, HTTP server, and domain controller when the HTTP client is using TLS authentication can be illustrated as follows.

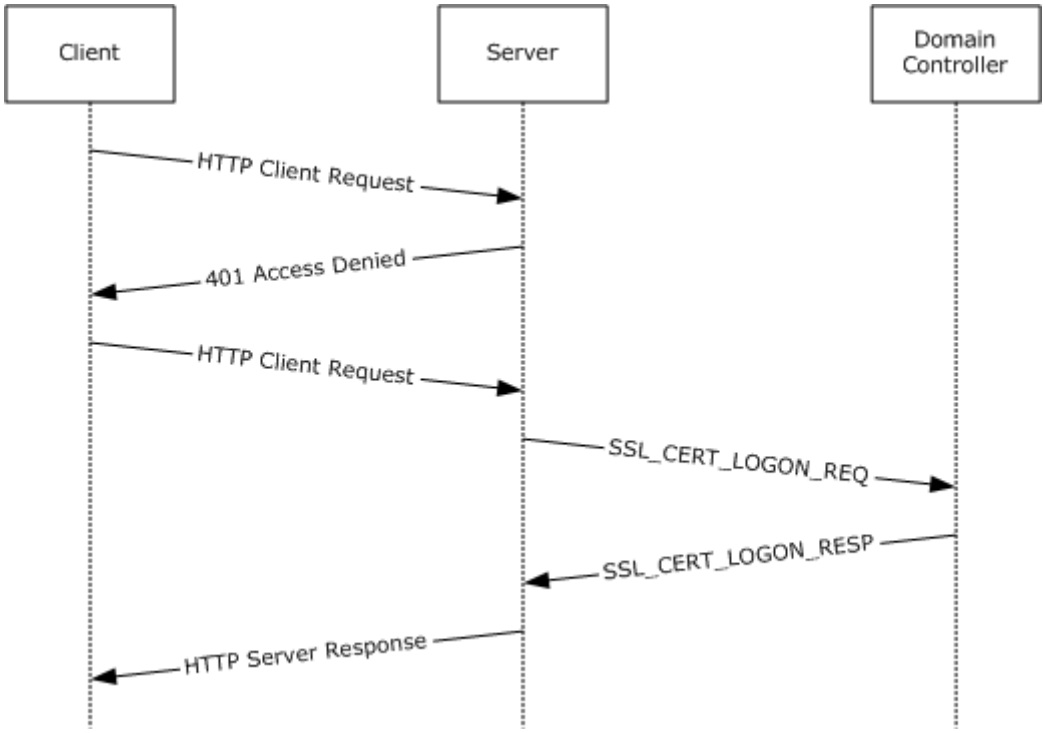

Figure 21: Protocol traffic for TLS authentication

5.4.5 Task Processing Rule Details

After the server's system startup and domain join have been completed as summarized in section 5.4.1, the server can initiate an HTTP Access Authentication - Server task.

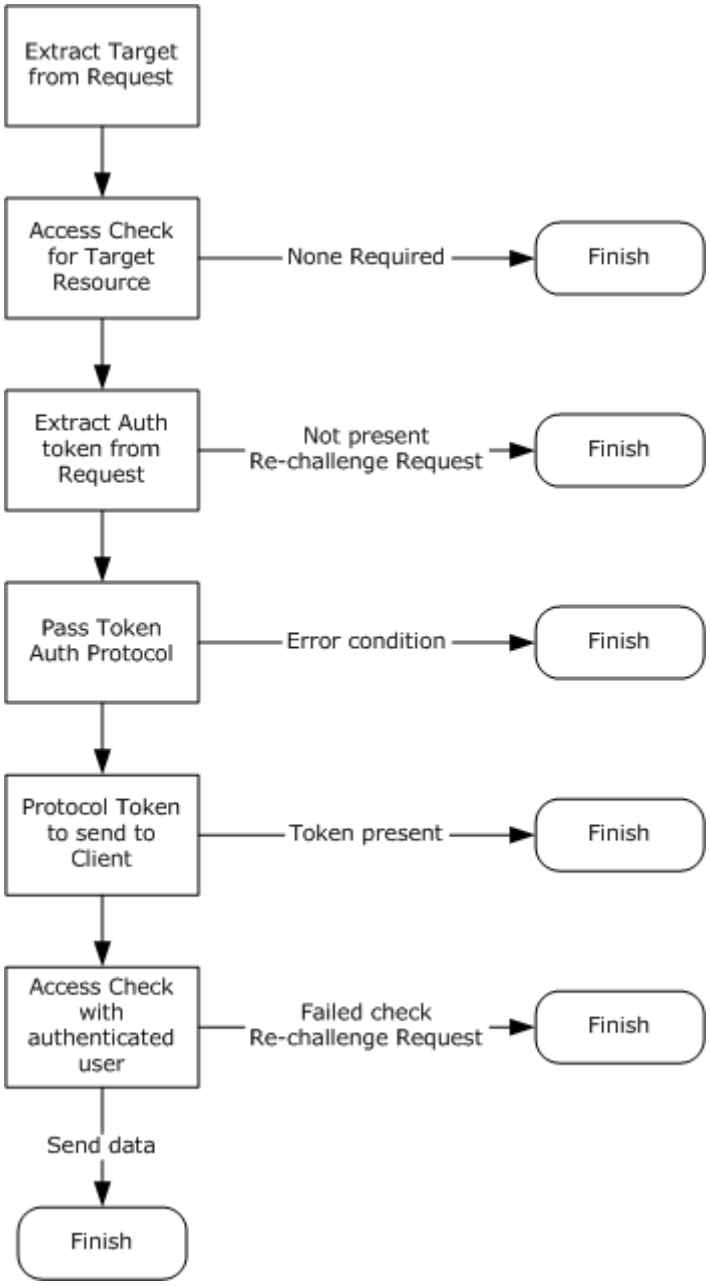

Figure 22: Flow diagram for the HTTP Access Authentication - Server task

The detailed sequence is as follows:

Step 1: Server extracts requested resource target name (RequestedResource) from the HTTP request. The parsing routines are implementation details, but MUST follow [RFC2616].

Step 2: Server determines if authenticated access is required for the requested resource. The access check is an implementation detail and may differ between various implementations.

Step 3: If no authentication is required for access, return Success and exit the HTTP Access Authentication – Server task. The HTTP service does not require authentication for the specified resource, RequestedResource.

Step 4: Service extracts the authentication information ClientAuthMechanism (initialized as auth-scheme in [RFC2617]) and ClientAuthToken (initialized as auth-param in [RFC2617]) that is attached to the HTTP request. The service parses out the authentication token that was transmitted on behalf of the client user by the HTTP client. The parser is an implementation detail and may differ between various implementations.

Step 5: If there is no authentication information (as is the case with anonymous requests), go to Step 12. Since Step 3 determined that authentication was required for access to the specified resource, RequestedResource, the HTTP request MUST fail due to lack of request authentication.

Step 6: Service determines which authentication protocol (ClientAuthProtocol) is being used in authentication mechanism extracted in Step 4. The ClientAuthProtocol MUST be one of the mechanisms contained in the configured list, ServerAuthenticationMechanism; if not, the HTTP server MUST fail the request.

Step 7: Based on the mechanism specified in ClientAuthMechanism, the service sends the authentication information to the specified authentication protocol. The specific mapping of authentication mechanism to authentication protocol is as follows (and is specified in the RFCs and TDs):

- "Digest" maps to DPSP [MS-DPSP]
- "NTLM" maps to NLMP [MS-NLMP]
- "Negotiate" maps to SPNG [MS-SPNG]
- "Nego2" maps to SPNG [MS-SPNG] conforming to [MS-N2HT].<2>

The protocol described in [MS-SPNG] can negotiate between multiple protocols. Windows offers KILE and NLMP as authentication protocols within SPNG.

Within these protocols on the server, the APDS protocol is used to communicate with the domain controller. For each of these protocols, APDS inputs are extracted from ClientAuthToken as specified in [RFC2617], [RFC4559], and [MS-N2HT]. The following diagram illustrates the components associated with the protocol described in [MS-APDS].

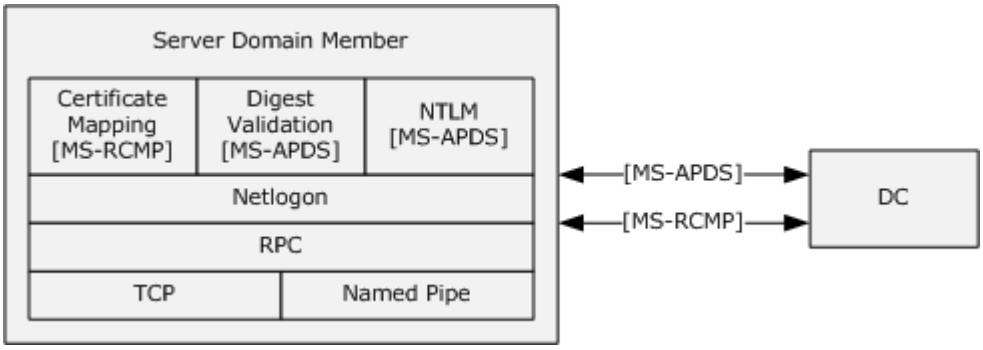

Figure 23: Components of the protocol described in [MS-APDS]

Step 8: If the authentication protocol returns an error, go to Step 12.

Step 9: If the authentication protocol returns that additional data is required, the HTTP response will be sent to the client, and the task is completed. If the protocol returns a token to pass back to the client (by means of the HTTP response), the token information is encoded as specified in [RFC2617], [RFC4559], and [MS-N2HT]. If success is returned from the authentication protocol, a logon token, UserLogonToken, will be created and is to be used in processing the HTTP request.

Step 10: HTTP service verifies authenticated response with access check for resource. The details on how an access check is accomplished based on the PAC returned in Step 7 is an implementation detail. The PAC conforms to [MS-PAC].

Step 11: If the access check is successful, return success and exit the HTTP Access Authentication – Server task.

Step 12: HTTP server generates authentication challenge to send back to client and returns error Access_Denied. This step is the result of a failed access check or an error raised during the previous steps. The HTTP server will return access denied status and re-challenge the client as per [RFC2616].

5.5 Task Security

This section documents security issues specific to this task that are not otherwise described in the Technical Documents (TDs) for the protocols used in the task. It does not duplicate what is already in the protocol TDs unless there is some unique aspect that applies to the system as a whole.

The logon token, UserLogonToken, that was created within the LSA based on the credentials provided by the user can be used for any process requests associated with the HTTP session. Care is taken to protect this logon token from other processes or HTTP sessions that are not associated with the user authentication.

Account lockout can be employed using the results of the authentication check of the user's credentials. If domain policy is configured with maximum bad password count and lockout duration, the results from the KRB_AS_REQ or APDS protocol can be controlled to return an error when the maximum bad password count has been reached.

6 File System Services Authentication - SMB Server Task

This section describes File System Services Authentication - SMB Server task. Server Message Block (SMB) protocols are detailed in [MS-SMB] and [MS-SMB2].

The Server Message Block Version 1.0 Protocol [MS-SMB] extends the Common Internet File System (CIFS) Protocol [MS-CIFS] and provides enhanced security, file, and disk management support. SMB is a stateful protocol, where clients establish a session with a server and use that session for a variety of requests to access files, printers, and interprocess communication (IPC) mechanisms such as named pipes. CIFS imposes state to maintain an authentication context, cryptographic operations, file semantics such as locking, and similar features.

The SMB extensions do not alter the basic message sequencing of the CIFS protocol but introduce new flags, extended requests and responses, and new information levels. All of these extensions follow a request/response pattern in which the client initiates all requests. The base protocol allows for one exception to this request/response pattern, oplock breaks, which are detailed in [MS-CIFS] section 3.2.4.18.

The Server Message Block Version 2 Protocol [MS-SMB2] is a major revision from the original SMB protocol. The packet formats are completely different from those of the original SMB protocol; however, many of the underlying concepts are carried over. To retain compatibility with existing clients and servers, the existing SMB protocol can be used to negotiate the use of the SMB Version 2 Protocol, as described in [MS-SMB2] section 1.7. After a dialect of the SMB2 protocol is selected during SMB negotiation, all messages that are sent on the connection (including the SMB negotiate response) will conform to SMB Version 2 Protocol messages, and no further SMB traffic will be exchanged on the connection.

6.1 Task Overview

6.1.1 Task Purpose

This task describes the steps that the server undertakes to provide support for authentication of SMB file system services. To enforce access controls over files and resources on an SMB server, the server MUST know the validated identity of the requestor. SMB provides support for several authentication protocols and the ability to negotiate authentication between the client and server.

Compared to authentication support provided by CIFS, SMB provides new authentication methods, including Kerberos. The SMB Negotiate and SMB Session Setup commands have been enhanced to carry opaque security tokens to support mechanisms compatible with the Generic Security Services (GSS) [RFC2743]. SMB and SMB2 rely on the Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO), which is specified in [RFC4178] and [MS-SPNG], for authentication, which in turn relies on Kerberos [MS-KILE] and the NTLM [MS-NLMP] challenge/response authentication protocol.

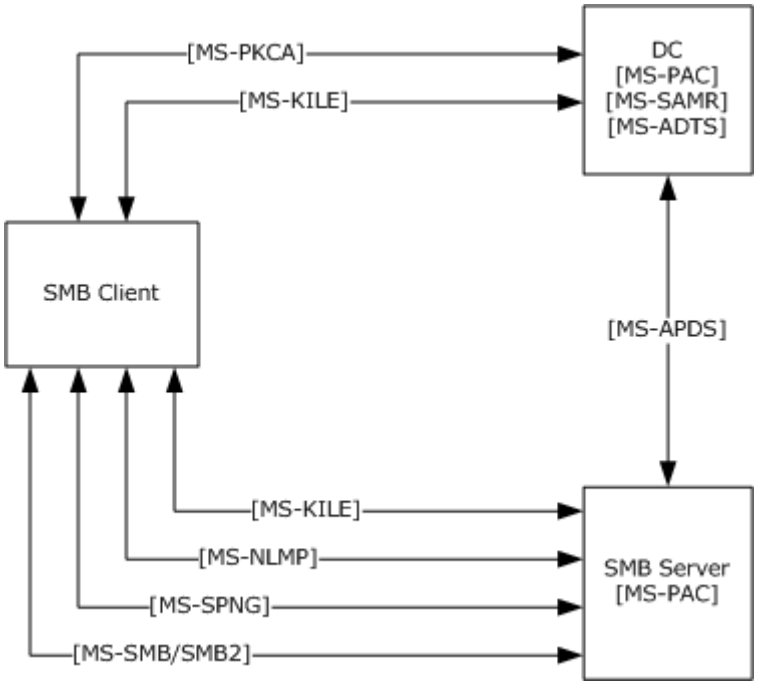

Figure 24: Standards used by the File System Services Authentication - SMB Server task

The diagram shows that SMB network traffic conforms to the SMB or SMB2 protocols used between the SMB client and the SMB server. The SMB server authenticates the user credentials provided by the SMB client using the APDS protocol to the domain controller that contains the user's account information or, if Kerberos is used for the SMB authentication, through KILE in obtaining a service ticket to the RPC service.

6.1.2 Task Applicability

This task is applicable to providing access controls over server system resources that are made available via the SMB protocols.

6.1.3 Task Use Cases

6.1.3.1 Stakeholders and Interests Summary

The stakeholders for this task are:

Server - The server is any server other than a domain controller server that is a member of the domain and is offering SMB services to clients in the domain. In practice, the domain controller can be the server, but for protocol evaluation, the domain controller and server will be considered separate.

Client - As shown in the preceding diagram, the client originates the SMB requests. As part of this task, the client will also be processing the user credentials and responding to the authentication requests from the SMB server.

6.1.3.2 Supporting Actors and Task Interests Summary

The supporting actors for this task are:

Administrator - Same as described in section 4.1.3.2. For this task, this entity will provision the domain credentials for the user to accomplish the task of access to SMB resources.

Server Administrator - The server administrator is responsible for the server itself. The server administrator manages the server and determines the resources that are managed by the server, and the associated policies for those resources. The server administrator relies upon the domain (and domain controller) to publish necessary rendezvous information for clients to find the server, authenticate the clients, and provide the necessary authorization information for the server to manage its resources. Depending on the server, publication of information may or may not be required. For this task, this entity will provision the SMB server to require authentication for local resources to be made available to the SMB service.

User - This entity is described in section 4.1.3.1. For this task, this user initiates the process on the SMB client. The task on the SMB server is being performed at the request of the user to gain access to the SMB resource server for resources on behalf of the user.

Domain Controller - As described in section 4.1.3.2. For this task, this domain computer provides validation of the user credentials, and upon successful validation, also provides user-specific authorization information. The APDS protocol is used by the SMB server.

6.1.3.3 Use Case Diagrams

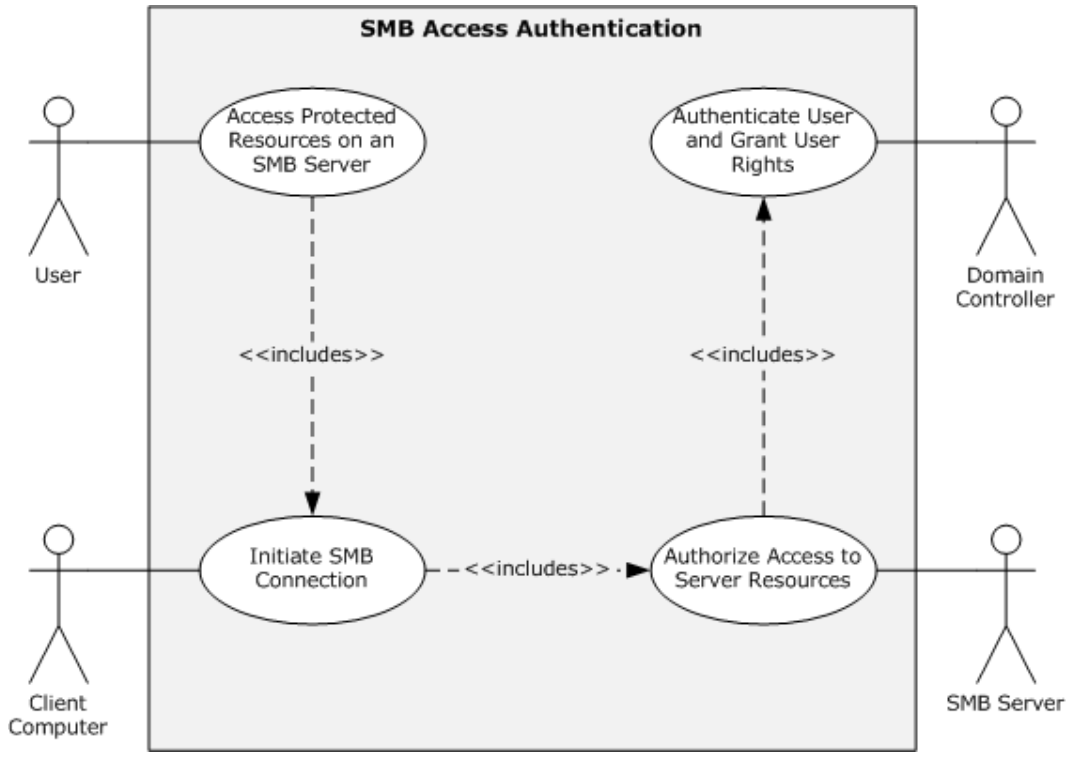

Figure 25: File System Services Authentication - SMB Server use case diagram

Here the user is initiating an SMB connection from the SMB client that is running on the client computer. The client computer may or may not be joined to the domain that is administered by the domain controller. The server computer is running the SMB server service. The administration of the SMB server may be done locally by the server administrator, or through policy by the domain administrator through the domain services.

The server administrator will provision the SMB server on the server computer. The administrator will determine which resources (files) would need authorization to access and set the appropriate SMB server configuration options to require authentication. The administrator can also configure which forms of authentication protocols are acceptable.

The domain administrator also needs to provision the user accounts. Part of this provisioning is to grant the user-specific rights that are appropriate for the user. These rights will be sent over to the SMB server on successful authentication through either the APDS protocol or as part of the KILE service ticket and evaluated locally against the requirements placed on the server resources.

6.1.3.4 SMB Access Authentication - Server

For the Use Case of File System Services - SMB Server, the user is initiating the SMB request with the goal of access to protected resources on an SMB server. The task focuses on the server-side steps that are triggered by the arrival of the SMB request. The server will be preconfigured for acceptable authentication mechanisms. The SMB server will deny unauthorized access to protected resources. To gain access, the user MUST provide credential material to the SMB client, which will be used to authenticate the SMB requests to the SMB server.

Goal: The user needs to access protected resources on an SMB server.

Context of Use: This is typically done when the user opens a file share.

Direct Actor: The user is the Direct Actor.

Primary Actor: Same as the Direct Actor.

Supporting Actors: The Domain Administrator, Server Administrator, and Domain Controller are the Supporting Actors.

Stakeholders and Interests:

- The User who is attempting to access SMB resources.
- The Domain Controller that is available to authenticate the user.

Precondition:

- The domain controller is available.
- The user account is provisioned.

See section 6.2.3 for detailed information.

Minimal Guarantees: The user will be authenticated or the authentication will fail.

Success Guarantee: The user is authenticated.

Trigger: The user attempts to open a file share.

Main Success Scenario:

1. The user opens a file share.
2. The explorer sends an SMB request to the SMB service.
3. The server sends an authentication request to the Domain Controller.

4. The Domain Controller authenticates the user.

Extensions: None.

6.2 Task Context

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

SMB and SMB2 protocols provide file, print services, and interprocess communication (IPC) mechanisms such as named pipes. These protocols are used by a number of other protocols and tasks including RPC [MS-RPCE] and Distributed File System (DFS) [MS-DFSC].

6.2.1 Task Environment

SMB extends the Common Internet File System (CIFS) [MS-CIFS]. CIFS provides an open cross-platform mechanism for clients to request services from a server computer that is reachable over a network. For file services, the protocol provides standard file operations such as open, close, seek, read, and write. CIFS also has support for record locking, caching (read-ahead and write-behind), protocol dialect negotiation, and extended file attributes. Authentication in CIFS is detailed in [MS-CIFS] section 3.1.5.2 and 2.2.4.52.

SMB extends the CIFS protocol by adding support in enhanced security, file management, and disk management. SMB follows the basic messaging sequencing of CIFS. Specifically, SMB adds protocol support for negotiate (SPNEGO) [MS-SPNG] which includes NTLM [MS-NLMP] and Kerberos [MS-KILE] authentication. The Negotiate and Session Setup commands have been extended to carry opaque security tokens to support authentication mechanisms that are compatible with Generic Security Services (GSS). Details on the SMB protocol exchange for client-to-server authentication during SMB_COM_SESSION_SETUP_ANDX are provided in [MS-SMB] sections 3.2.4.2.4.1 and 4.1; and during SMB2_SESSION_SETUP are provided in [MS-SMB2] sections 2.2.6, 3.3.5.5, and 4.1. The details on the authentication mechanisms are provided in [MS-SMB] section 3.2.4.2.4, or for the SMB Version 2 Protocol, in [MS-SMB2] section 3.2.5.3.

For authentication, the SMB2 protocol relies on Simple and Protected GSS-API Negotiation (SPNEGO), as specified in [RFC4178] and [MS-SPNG], which in turn may rely on the Kerberos Protocol Extensions (as specified in [MS-KILE]) or the NTLM Authentication Protocol (as specified in [MS-NLMP]). The SMB Version 2 Protocol does not use direct NTLM [MS-NLMP], unlike SMB.

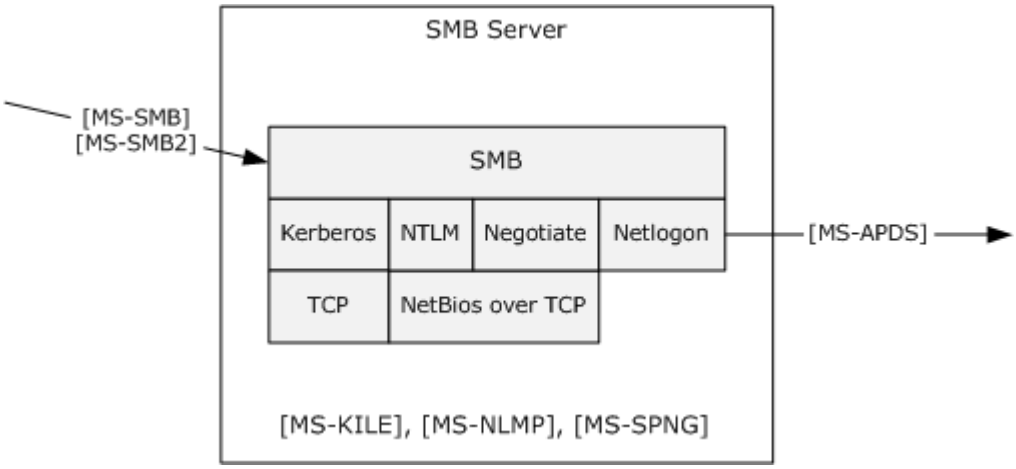

Figure 26: Components of SMB Server

SMB uses direct TCP, NetBIOS over IPX, NetBIOS Extended User Interface (NetBEUI), or NetBIOS over TCP as underlying transports. SMB2 uses either direct TCP or NetBIOS over TCP as underlying transports.

6.2.2 Task Relationships

6.2.2.1 Black Box Relationship Diagrams

The File System Services Authentication - SMB Server task uses a number of protocols (some are prerequisite and others are used within the task itself). The SMB client triggers the task - specifically, the client directs the SMB client to communicate an SMB request to the SMB server. The task involves a server member computer in the domain as well as the authority for the domain, that is, the domain controller. The SMB client may be either a domain-joined computer or a non-domain-joined computer.

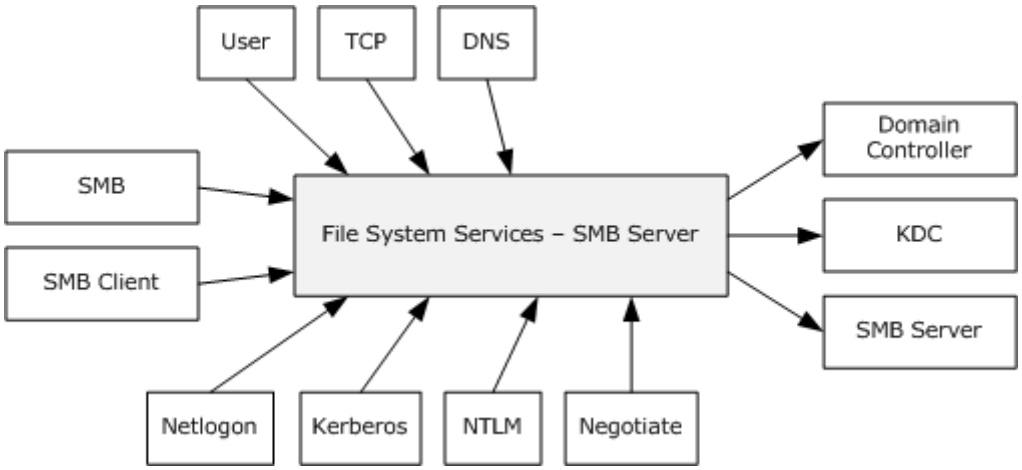

Figure 27: Black box relationships for the File System Services Authentication - SMB Server task

6.2.2.2 Task Dependencies

This task has a number of external dependencies. The following protocols are used by this task:

- Precondition - System Level
  - Domain Interaction System Overview [MS-DISO]
- Precondition - Protocols
  - TCP/IP
  - DNS
  - SMB (either SMB or SMB Version 2)
  - Negotiate
  - NTLM
  - Kerberos

- Netlogon
- Group Policy

- Precondition - Tasks
  - Server Domain Join
- External Server System
  - Active Directory
  - DNS Directory
  - KDC

6.2.2.3 Task Influences

The task has several external influences. These influences can alter state of the protocols, for example limiting which authentication mechanisms are offered.

External influences are as follows:

- Group Policy for the user account for the domain logon
- Active Directory- account lockout subsystem, account restrictions
- Security event logging on the domain controller
- SMB resource access requirements (authorization)

6.2.3 Task Assumptions and Preconditions

This task has the precondition that the domain controller has been configured to support the domain infrastructure. The user account for the client authentication has been created and provisioned on the domain controller.

Also, as part of the Windows startup procedure for the server computer, the following actions have taken place (see [MS-DISO] for additional details):

- Successful connection to the network and startup of networking stacks.
- Successful domain join.
- Establishment of a secure channel with the domain controller.
- Startup and configuration of the SMB service.

6.2.4 Task Versioning and Capability Negotiation

SMB supports negotiation for authentication. This task uses the versioning and negotiation support provided by the respective protocols used for this task. See each Protocol Technical Document for details on a specific protocol used. This task does not have specific support for Task Versioning and capability negotiation beyond the TDs referenced.

6.3 Task Architecture

The basic architecture contains a domain controller, a member server computer, and a client computer. The SMB client is triggering a File System Services Authentication - SMB Server task for the server resources.

6.3.1 Task Architectural Constraints

The primary constraint is that the server computer has successfully accomplished a domain join and is operating within the network boundaries of the domain. The member machines MUST have network communication access to the domain controller. The SMB client MUST have network access to the SMB server.

6.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by the processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

ServerSecurityContext - A security context that represents the server system.

6.3.3 Task Abstract Parameters

This section describes data that is passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

LogonServer - The name of the domain controller to be used for authentication.

DNSServer - The name of the server to be used for DNS queries.

6.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

UserLogonToken - A logon token created on the SMB server during user authentication of an SMB request.

6.3.5 White-Box Relationships

The following figure represents the white-box relationships of the task within the client computer, server computer and with the domain controller.

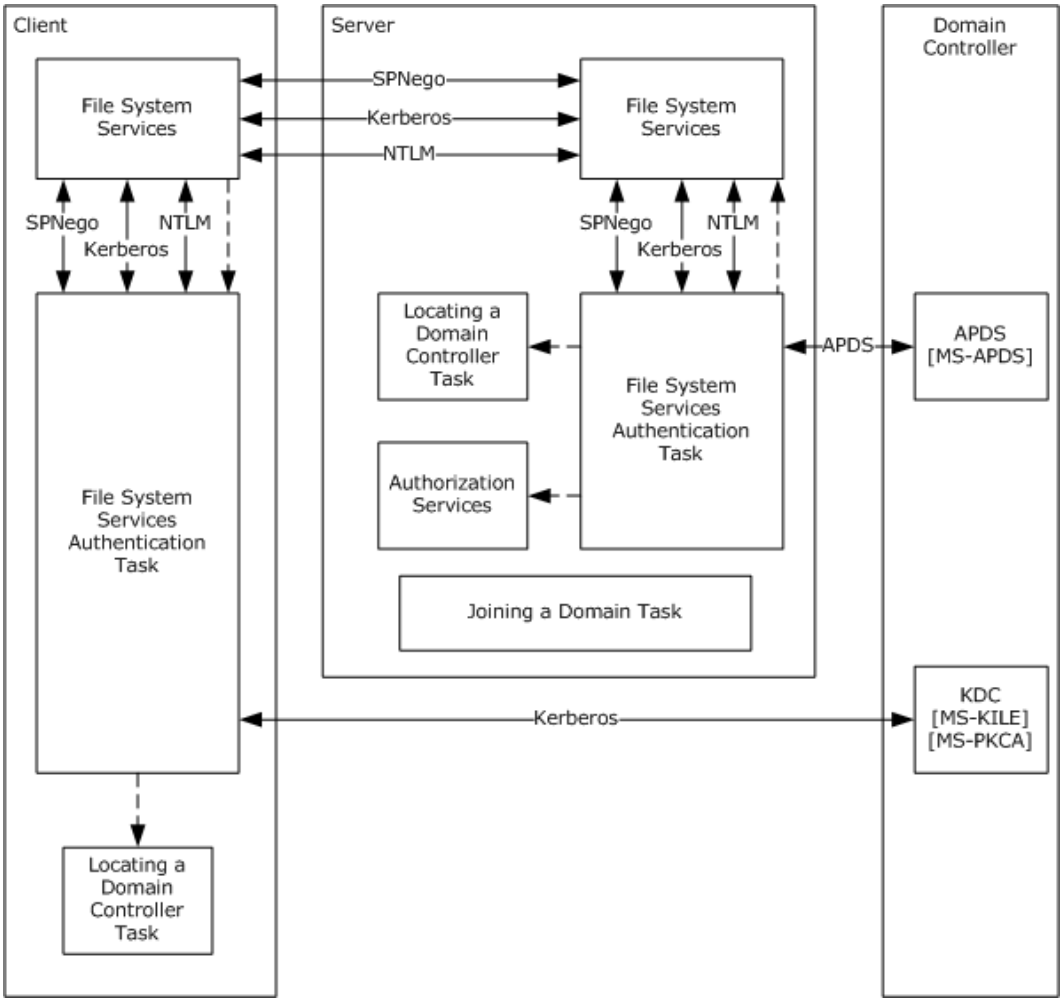

Figure 28: White box relationships for the File System Services Authentication - SMB Server task

6.3.6 Task Events

This task initiates audit events based on the success/failure of the task. The authentication audit events are logged at the domain controller for the verification of the user credentials. Authentication audit events and access check events are also logged at the server computer.

6.3.6.1 Task Timers

There are no timers employed directly by this task. Timers may be employed by invoked protocols (such as KILE). Invoked protocol timers are detailed in their respective Protocol Technical Document.

6.3.6.2 Task Non-Timer Events

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

6.3.7 Task Architecture and Communication

In SMB, enhanced security is indicated by the SMB_FLAGS2_EXTENDED_SECURITY bit, as detailed in [MS-SMB] section 2.2.3.1. The SMB server, by means of the SMB_COM_NEGOTIATE server Response Extension ([MS-SMB] section 2.2.4.5.2.1), will indicate that it supports extended security by setting the CAP_EXTENDED_SECURITY bit in the Capabilities bit field.

To support authentication token exchange between the server and the client, the authentication protocol's token is passed as the SecurityBlob in the SMB_COM_NEGOTIATE Server Response Extension and in SMB_COM_SESSOIN_SETUP_ANDX. Details on the SMB protocol exchange for client-to-server authentication during SMB_COM_SESSION_SETUP_ANDX are provided in [MS-SMB] sections 3.2.4.2.4.1 and 4.1. The details on the authentication mechanisms are provided in [MS-SMB] section 3.2.4.2.4.

The diagram below details the SMB protocol traffic between the SMB client and the SMB server, using SMB_COM_SESSION_SETUP_ANDX for establishing the user level authentication.

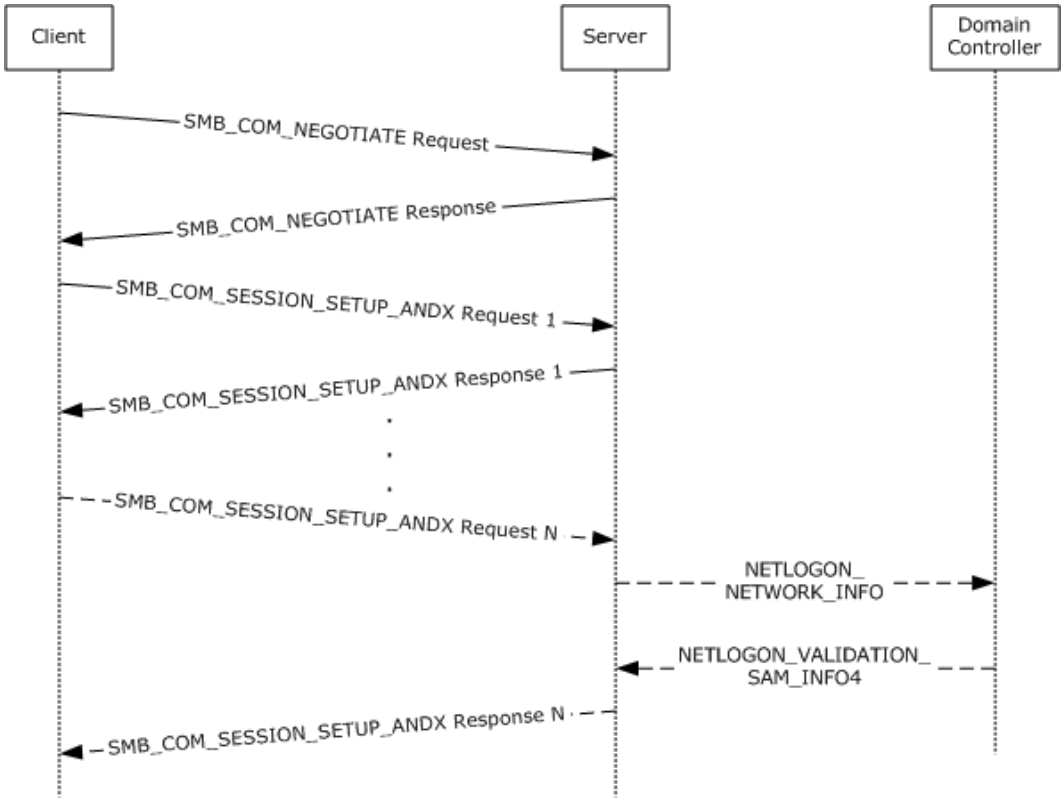

Figure 29: Sequence diagram for SMB protocol traffic

To validate the NTLM challenge material in the SMB client's SMB_COM_SESSION_SETUP_ANDX request, the SMB server uses NTLM [MS-NLMP], which in turn uses APDS [MS-APDS] as shown in the preceding diagram.

To validate a Kerberos ticket in the SMB client's SMB_COM_SESSION_SETUP_ANDX request, the SMB server uses its Kerberos key.

In a similar fashion, the SMB Version 2 Protocol [MS-SMB2] usage can be illustrated as follows.

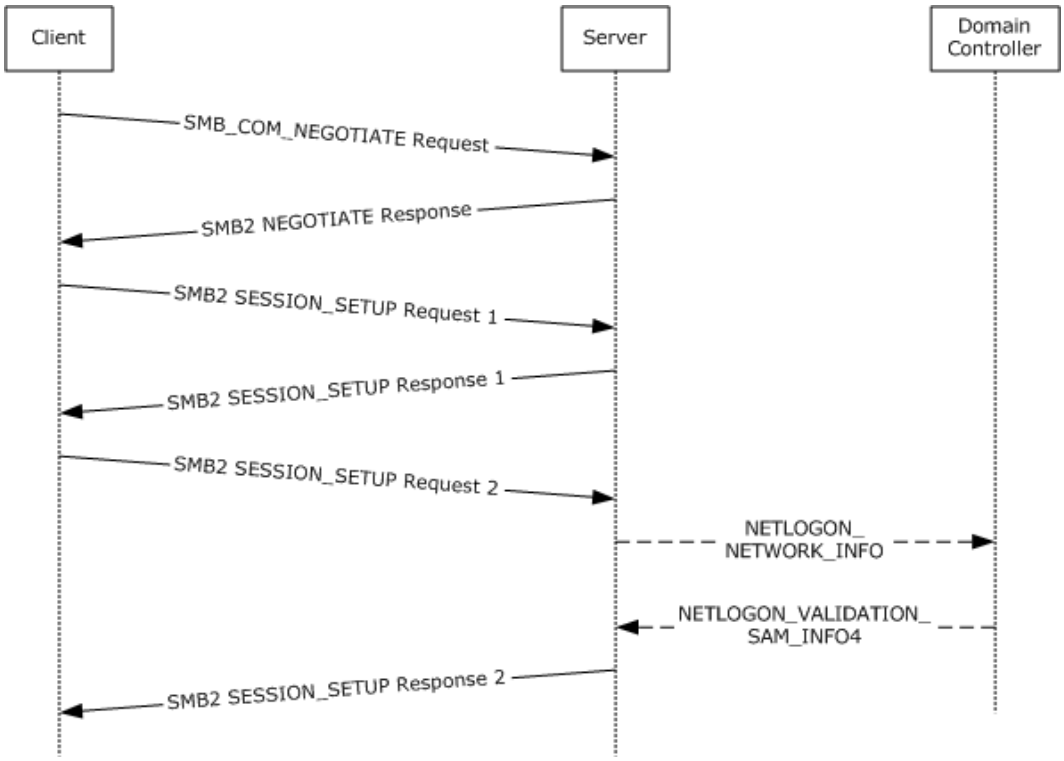

Figure 30: Sequence diagram for SMB2 protocol traffic

6.3.8 Task Processing Rules

Init SMBAccessAuth:

Step 1: Check to see if the server computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

Step 3: The server computer starts up the SMB service.

Step 4: The user provides credential information to the SMB client for authenticating SMB requests to the remote SMB server.

Step 5: The client computer sends an SMB request to the server.

Step 6: The server is waiting for client requests, and the server receives the SMB request from the client and passes it to the SMB server.

Task SMBAccessAuth:

Step 1: The SMB server evaluates the level of authentication specified by the SMB client.

Step 2: If the client provided unacceptable authentication parameters, the server closes the connection and goes to "Init SMBAccessAuth:Step 6", shown above, where it waits for client requests.

Step 3: The server produces a response containing initial authentication data for the client. The initial authentication data is generated by a server-supported authentication protocol.

Step 4: The client and server continue to exchange SMB messages until the server authentication protocol determines that it has sufficient information to attempt to authenticate the connection requests.

Step 5: The server authentication protocol contacts the domain controller to validate the credential information.

Step 6: If the domain controller validates the credentials, the SMB server returns a successful authentication response; otherwise, the SMB server returns a failure status to the SMB client.

Each of the primary steps within this task can return errors. Appropriate error handling SHOULD be done.

6.3.9 Task Failure Scenarios

Each of the primary steps in the File System Services Authentication - SMB Server task can result in a failure case. On any error not directly specified in the processing tasks, the task MUST return a Sytem_Error error and fail the SMB request.

Some of the protocols contain recoverable errors that are handled in the lower level protocols. For example, in the KILE protocol, the KDC will be contacted during Step 5. The KILE protocol will retry contacting the KDC up to three times if there is no response. This failure/recovery path is handled at a lower protocol level and is not directly influenced by the task.

6.4 Task Details

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

6.4.1 Task Precondition Details

This task has a precondition that the domain controller has been configured to support the domain infrastructure. The user account for the client member logon has been created and provisioned on the domain controller. Networking support has been established between the client, server, and domain controller.

Also, as part of the Windows startup procedure for the server, the actions specified in section 4.4.1 have taken place (see [MS-DISO] for additional details). The server has successfully joined a domain.

Step 1: Check to see if the server computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

The following additional steps will be undertaken:

Step 3: The server computer starts up the SMB service. The server is required to run a service to support SMB protocol server-side operations as specified in [MS-SMB] and [MS-SMB2].

Step 4: The user provides credential information to the SMB client for use in authenticating SMB requests to the remote SMB server.

Step 5: The client computer sends an SMB request to the server. The client computer initiates an SMB request conforming to [MS-SMB] or [MS-SMB2] on behalf of the user. SMB and SMB2 require that a client that initiates a CIFS connection negotiate with an SMB dialect of NTLM 0.12 (as specified in [MS-CIFS] section 1.7).

Step 6: The server is waiting for client requests, and the server receives the SMB_COM_NEGOTIATE client request and passes it to the SMB server. If the SMB_FLAGS2_EXTENDED_SECURITY bit is set in the client header, the client supports extended security negotiation.

6.4.2 Task Initialization of External Entities

This task does not initialize external entities. The external protocols used by this task have been initialized during system startup and during the domain join task.

6.4.3 Task Event Details

6.4.3.1 Task Timer Details

There are no timers employed directly by this task. Timers may be employed by invoked protocols (such as MS-KILE). Invoked protocol timers are detailed in their respective Protocol Technical Document.

6.4.3.2 Task Non-Timer Event Details

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

6.4.4 Task Architectural Details

The following figure illustrates the steps that make up the task in a flow diagram.

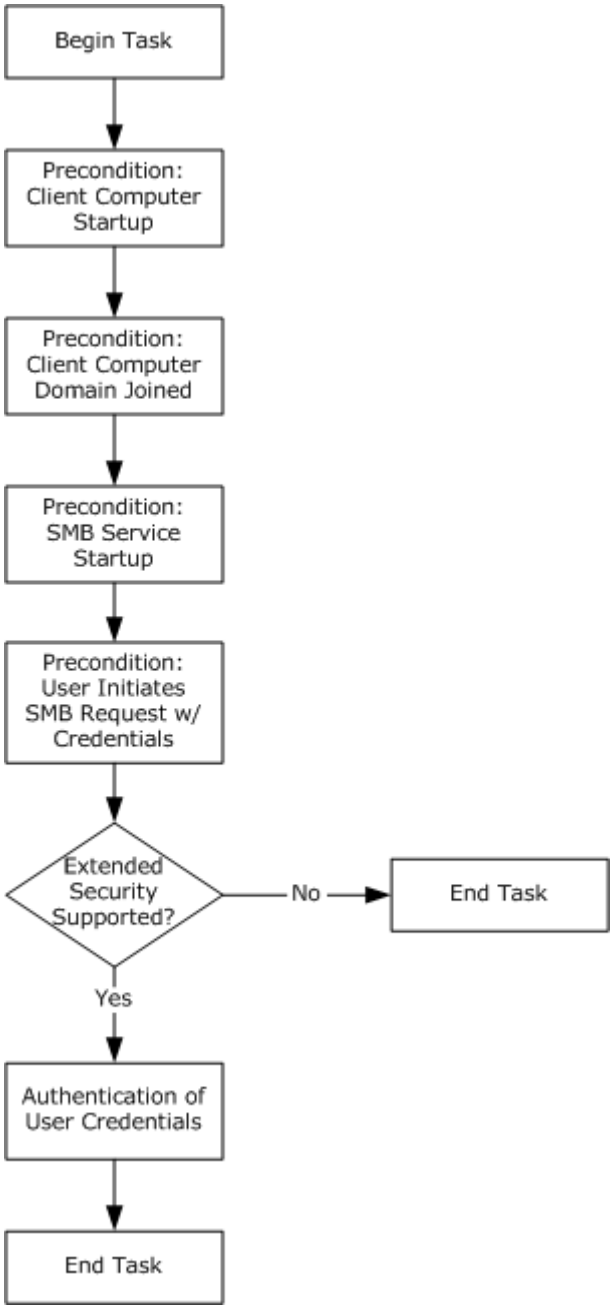

Figure 31: Flow diagram for the File System Services Authentication - SMB Server task

The following figure illustrates the protocols that can be used in this task. The protocols have been initialized during server computer system startup and during domain join; specifically, Kerberos has detected the KDC for the domain.

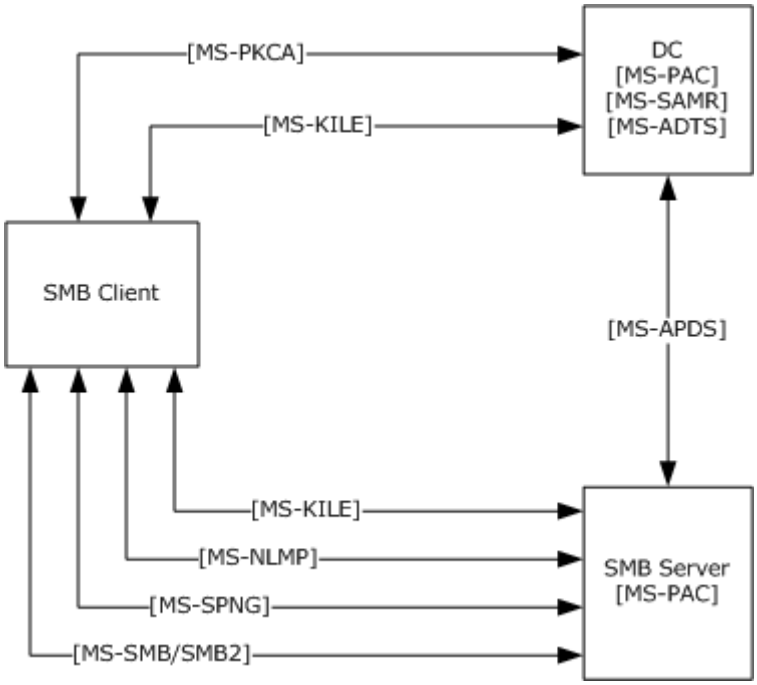

Figure 32: Standards used by the File System Services Authentication - SMB Server task

The client and server can negotiate for one of several authentication mechanisms according to [MS-SMB] and [MS-SMB2]. The SMB protocols can use NTLM or Negotiate, which in turn may select from Kerberos or NTLM. If Kerberos is negotiated, the KILE protocol will be used by the client to contact the domain controller to obtain a ticket. The client provides the ticket to the server via KILE encapsulated within the Negotiate Protocol with the SMB request.

The authentication token information is contained within the SMB request and response according to the authentication standards and TDs.

For both SMB and SMB Version 2 Protocols, the SMB client and server MUST call the GSS authentication protocol with the MutualAuth and Delegate options. In addition, for SMB Version 2 the GSS_C_FRAGMENT_TO_FIT parameter ([MS-SPNG] section 3.3.1) will be set.

The server, depending on the authentication mechanism used in the SMB request, will use [MS-APDS] section 1.4, as described in the TD.

6.4.5 Task Processing Rule Details

After the server's system startup and domain join have been completed as summarized in section 6.4.1, the SMB server can process the SMB authentication requests.

The detailed sequence is as follows:

Step 1: The SMB server evaluates the level of authentication specified by the SMB client. The SMB client MUST have specified and negotiated the NTLM 0.12 dialect. If the client supports extended security negotiation, GSS negotiation authentication, the SMB_FLAGS2_EXTENDED_SECURITY bit is set in the SMB header in the SMB_COMM_NEGOTIATE client request.

Step 2: If the client provided unacceptable authentication parameters, the server fails the connection and goes to Step 6 in section 6.4.1, where it waits for client requests. If the server is

configured to require GSS authentication and the CAP_EXTENDED_SECURITY bit in Capabilities in the SMB_COMM_SESSION_SETUP_ANDX client request is not set, the server can fail the connection.

Step 3: For SMB, the server produces an SMB_COMM_NEGOTIATE server response that contains initial authentication data for the client. The initial authentication data is generated by a server-supported authentication protocol. To use GSS negotiation authentication, the SMB server MUST indicate this capability to the SMB client by specifying the CAP_EXTENDED_SECURITY bit in Capabilities in SMB_COMM_NEGOTIATE server response and by setting the SMB_FLAGS2_EXTENDED_SECURITY in the SMB header Flags2 field.

For the SMB Version 2 Protocol, the server produces an SMB2_COMM_NEGOTIATE and sends this over to the SMB client. GSS extended security negotiation is supported on both the SMB client and SMB server.

Step 4: The client and server continue to exchange SMB messages until the server authentication protocols determine that the server has sufficient information to attempt to authenticate the connection requests.

For GSS authentication within the SMB protocol, the SMB_COM_SESSION_SETUP_ANDX client request command MUST be generated by the SMB client, and the SMB server MUST generate the SMB_COM_SESSION_SETUP_ANDX response for SMB. The client MUST set CAP_EXTENDED_SECURITY in the Capabilities field to indicate that the client requests extended security negotiation; and set the SMB_FLAGS2_EXTENDED_SECURITY in the SMB header Flags2 field to indicate that the client supports extended security negotiation; and set the SecurityBlob field to the GSS negotiate output token; and set the SecurityBlobLength to the token length.

For NTLM implicit authentication with SMB protocol, the CAP_EXTENDED_SECURITY bit in ServerCapabilities is not set in SMB_COM_SESSION_SETUP_ANDX. The server MUST construct an NTLM CHALLENGE_MESSAGE, as specified in [MS-NLMP], and set the parameters according to [MS-SMB] section 3.2.4.2.4. The client creates the appropriate AUTHENTICATE_MESSAGE to respond to the CHALLENGE_MESSAGE, as specified in [MS-NLMP], and returns the security data in SMB_COM_SESSION_SETUP_ANDX client request.

For the SMB Version 2 Protocol, the SMB2_SESSION_SETUP contains the GSS negotiate output token and length.

Step 5: The server authentication protocol contacts the domain controller to validate the credential information.

In processing the SMB_COM_SESSION_SETUP_ANDX client requests, the server calls into GSS negotiate if extended security negotiation is negotiated by the client and server. If NTLM implicit is used, the NTLM [MS-NLMP] protocol is used directly. At one point in the authentication exchange, the SPNG or NLMP protocol has sufficient authentication data and can validate the client's authentication/credentials. The APDS protocol is used for this purpose and contacts the domain controller to verify the information.

In processing the SMB2_SESSION_SETUP client requests, the server calls only into the GSS negotiate security provider [MS-SPNG], which in turn calls into APDS.

Step 6: If the domain controller validates the credentials, the SMB server can return a successful authentication response; otherwise the SMB server returns a failure status to the SMB client. On success, a logon token, UserLogonToken, is created within the security context, ServerSecurityContext, on the server.

SPNG can negotiate between multiple protocols. Windows offers KILE and NLMP as authentication protocols within SPNG.

With the NTLM protocol on the server, the APDS protocol is used to communicate with the domain controller. APDS inputs are extracted from the SecurityBlob contained in the SMB_COM_SESSION_SETUP_ANDX request for SMB or SMB2 SESSION_SETUP as specified in [MS-SMB2]. The diagram below illustrates the use of APDS from the SMB server to the domain controller.

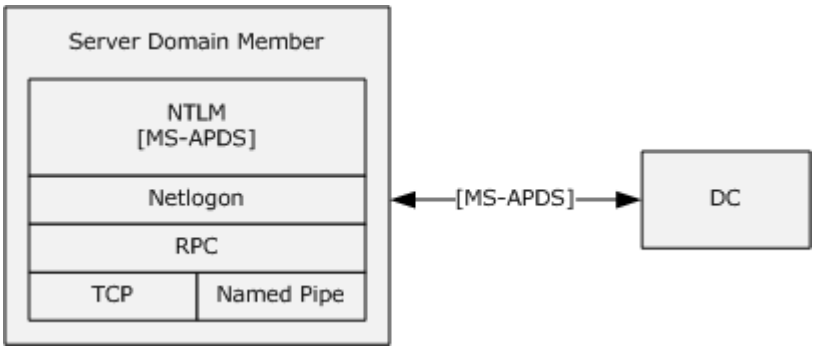

Figure 33: Components used by [MS-APDS] from the SMB server

6.5 Task Security

This section documents security issues specific to this task that are not otherwise described in the Technical Documents (TDs) for the protocols used in the task. It does not duplicate what is already in the protocol TDs unless there is some unique aspect that applies to the system as a whole.

The task security considerations are described in their respective Technical Documents.

7 Remote Procedure Services Authentication - RPC Server Task

This section describes the Remote Procedure Services Authentication - RPC Server task. Remote Procedure Call (RPC) protocols are detailed in [MS-RPCE] and for HTTP transport in [MS-RPCH]. RPC is a model for programming in a distributed computing environment and provides a form of inter-process communication. The goal of RPC is to provide transparent communication so that the client appears to be directly communicating with the server. The Windows implementation of RPC is compatible with the Open Software Foundation (OSF) Distributed Computing Environment (DCE) RPC.

With the increased use of distributed applications, the need for secure communications between the client and server portions of applications is critical. RPC provides a standardized interface to authentication services for both clients and servers. Server applications use authenticated remote procedure calls to ensure that all calls come from authorized clients.

[MS-RPCE] defines a set of extensions to the DCE 1.1: Remote Procedure Call (RPC) Specification, as specified in [C706]. These extensions add new capabilities to the DCE 1.1: RPC Specification, allow for more secure implementations to be built, and, in some cases, place additional restrictions on the DCE RPC Specification.

RPCE supports two different methods for adding security for distributed applications. The first method uses the system security protocols such as KILE and NLMP. In general, this is the preferred method to use and provides the most flexible and network-independent authentication features.

The second method is to use the security features built into the system transport protocols used by RPC. The transport-level security method is not the preferred method. Using the security protocols provided by RPC is recommended because it works on all transports, across platforms, and provides high levels of security, including privacy.

7.1 Task Overview

7.1.1 Task Purpose

This task describes the steps that the server undertakes to provide authentication for RPC system services. To complete any remote procedure call, all distributed applications will create a binding between the client and the server. To complete a secure remote procedure call, additional steps are necessary. Each authenticated client will provide authentication credentials to the server. Under RPC, the client stores its authentication credentials in the binding between the client and the server.

Before authenticated communication can take place between the client and server programs, the server will register a security provider with the local RPC runtime ([MS-RPCE] section 3.3.3.3.1.3). In particular, the server will register its service principal name if it supports Kerberos authentication and specify the authentication service it uses; that is, the server will choose a security provider (authentication service in DCE terminology) and decide on its authentication mechanism. After that, the client obtains a binding to the server, and requests a secure remote procedure call from the RPC runtime, and specifies various security options, such as security provider and security QOS options.

The RPC extensions that require message authentication and security rely on the following protocols: Kerberos [MS-KILE], Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO): Microsoft Extension [MS-SPNG] and [RFC4178], NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP], Net Logon Remote Protocol [MS-NRPC], and Authentication Protocol Domain Support Specification [MS-APDS].

RPCE uses a model with a pluggable security provider module for the actual security and authentication work. Higher-level protocols on the client can discover the security provider that is

supported by the server or know them in advance. Higher-level protocols on the client can negotiate the use of RPC security providers by sending a message using a given RPC security provider. If the server supports the RPC security provider, as specified in [MS-RPCE] sections 3.3.3.1, 3.2.3.5.4, and 3.3.3.5.3, the communication succeeds. If the server does not support the RPC security provider, the server returns an error, as specified in [MS-RPCE] section 3.3.3.5.3 for connection-oriented RPC protocols, or as specified in [MS-RPCE] section 3.2.3.5.4 for connectionless RPC protocols.

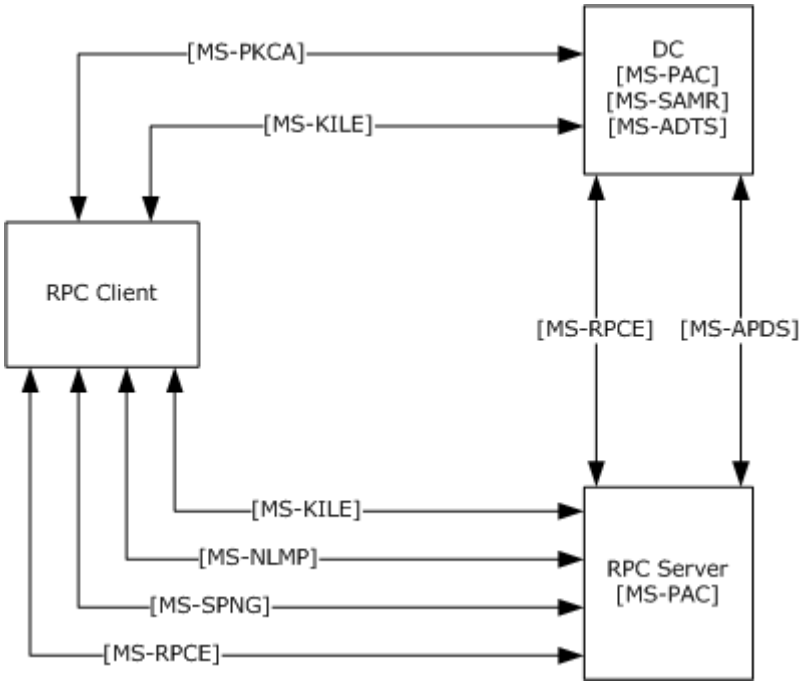

Figure 34: Standards used by Remote Procedure Services Authentication - RPC Server task

The diagram shows that RPC network traffic conforming to the RPCE protocol is used between the RPC client and the RPC server. If NTLM is used, the SMB server authenticates the user credentials provided by the RPC client through the APDS protocol to the domain controller which contains the user's account information or, if Kerberos is used for the RPC authentication, through KILE in obtaining a service ticket to the RPC service.

7.1.2 Task Applicability

This task is applicable to providing access controls over server system services and remote execution made available via the RPC protocol.

7.1.3 Task Use Cases

7.1.3.1 Stakeholders and Interests Summary

The stakeholders for this task are:

Server - The server is any server other than a domain controller server that is a member of the domain and offering RPC services to clients in the domain. In practice, the domain controller can be the server, but for protocol evaluation, the domain controller and server will be considered separate.

Client - As shown in the preceding diagram, the client originates the RPC requests. As part of this task, the client will also be processing the user credentials and providing the authentication data to the RPC server.

7.1.3.2 Supporting Actors and Task Interests Summary

The supporting actors for this task are:

Administrator - Same as described in section 4.1.3.2. For this task, this entity MUST provision the domain credentials for the user to accomplish task for access to RPC resources.

Server Administrator - The server administrator is responsible for the server itself. The server administrator manages the server and determines the resources that are managed by the server, and the associated policies for those resources. The server administrator relies upon the domain (and domain controller) to publish necessary rendezvous information for clients to find the server, authenticate the clients, and provide the necessary authorization information for the server to manage its resources. Depending on the server, publication of information may or may not be required. For this task, this entity MUST provision the RPC server to require authentication for local endpoints to be made available to the RPC service.

User - This entity is described in section 4.1.3.1. For this task, this entity initiates the action on the RPC client. The task is being performed at the request of the user to execute procedures on the RPC resource server computer as authenticated by the user credentials.

Domain Controller - As described in section 4.1.3.2. For this task, this domain computer provides validation of the user authentication, and upon successful validation, also provides user-specific authorization information.

7.1.3.3 Use Case Diagrams

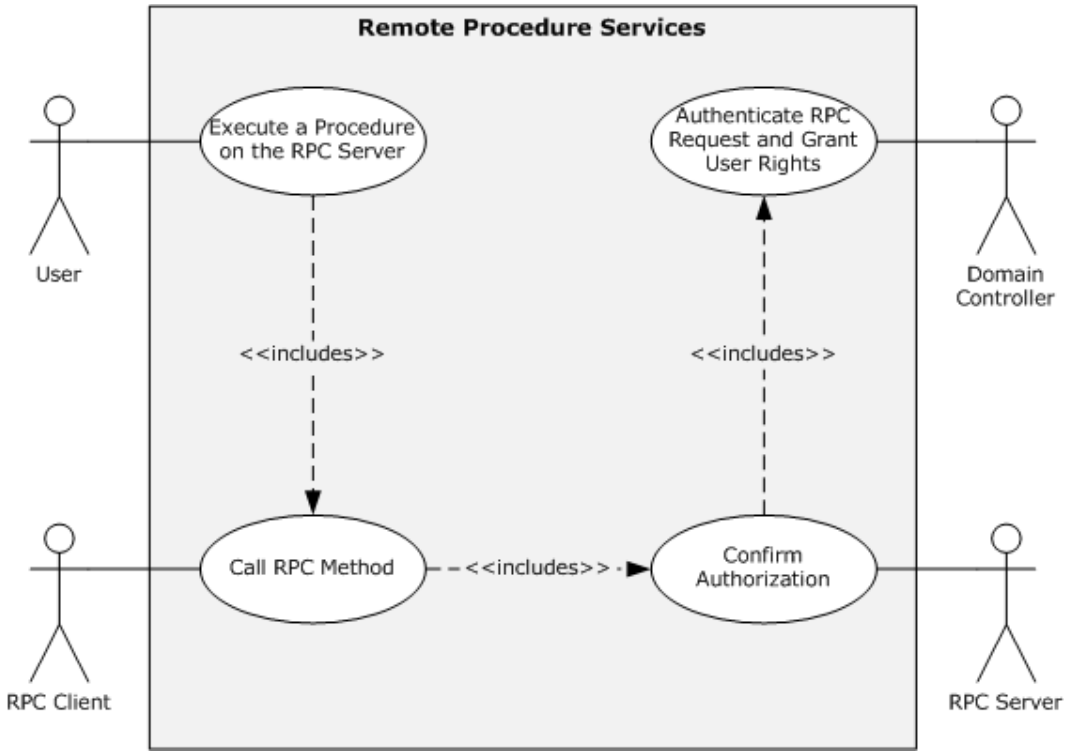

Figure 35: Remote Procedure Services Authentication - RPC Server use case diagram

Here the user is initiating an RPC connection from the RPC client running on the client computer. The client computer MAY be joined to the domain administered by the domain controller. The server computer is running the RPC server service. The administration of the RPC server may be done locally by the server administrator or through policy by the domain administrator through the domain services.

The server administrator will provision the RPC server on the server computer. The administrator would determine the authorization required and set the appropriate RPC server configuration options to require authentication via a security provider. The specific security provider mechanism is registered for the RPC service before any RPC client attempts to connect. The RPC client MUST use the same security provider as registered by the RPC server when making an RPC request.

The domain administrator also needs to provision the user accounts. Part of this provisioning is to grant the user specific rights that are appropriate for the user. These rights will be sent over to the RPC server on successful authentication through the APDS protocol or as part of the KILE service ticket and evaluated locally against the requirements placed on the server resources.

7.1.3.4 Remote Procedure Services - RPC Server

For the Use Case of Remote Procedure Services - RPC Server, the user is initiating the task with the goal of executing a procedure on an RPC server with specified parameters and receiving the output from that procedure. The server is configured with the authentication protocols that are allowed to be used. The RPC server will deny access to requests with unauthorized access to protected RPC resources. To gain access, the user will provide credential material to the RPC client to authenticate the RPC requests to the RPC server. If the user does not have authorization, the method will fail with access denied.

Goal: The user needs to execute a procedure on an RPC server.

Context of Use: This is typically done when the user runs an application that calls an RPC method.

Direct Actor: The user is the Direct Actor.

Primary Actor: Same as the Direct Actor.

Supporting Actors: The Domain Administrator, Server Administrator, and Domain Controller are the Supporting Actors.

Stakeholders and Interests:

- The User who is attempting to runs an application that calls an RPC method.
- The Domain Controller that is available to authenticate the user.

Precondition:

- The domain controller is available.
- The user account is provisioned.

See section 7.2.3 for detailed information.

Minimal Guarantees: The user will be authenticated or the authentication will fail.

Success Guarantee: The user is authenticated.

Trigger: The user attempts to run an application that calls an RPC method.

Main Success Scenario:

1. The user runs an application.
2. The application calls an RPC method for the RPC service.
3. The server sends an authentication request to the Domain Controller.
4. The Domain Controller authenticates the user.
5. The RPC service confirms user authorization for the RPC method.
6. The application call succeeds.

Extensions: None.

7.2 Task Context

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

To make a secure RPC call, a security context needs to be created before it can be used. Several protocols participate in establishing the security context between the client and the server. The process of creating a context involves exchanging one or more messages between the client and server implementations of a security provider. During this process, a security provider may optionally exchange messages with an entity other than the client or server. Such exchanges include messages to a Key Distribution Center (KDC) that uses the KILE protocol or to a domain controller through the APDS protocol.

7.2.1 Task Environment

[MS-RPCE] defines a set of extensions to the DCE 1.1: Remote Procedure Call (RPC) Specification, as specified in [C706]. RPC provides for inter-process communication and provides a widely used technology for creating distributed client/server programs.

For authentication, the RPCE protocol supports a number of security providers including the NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP] and Simple and Protected GSS-API Negotiation (SPNEGO) [RFC4178] [MS-SPNG] which in turn may rely on the Kerberos Protocol Extensions [MS-KILE] or NTLM [MS-NLMP].

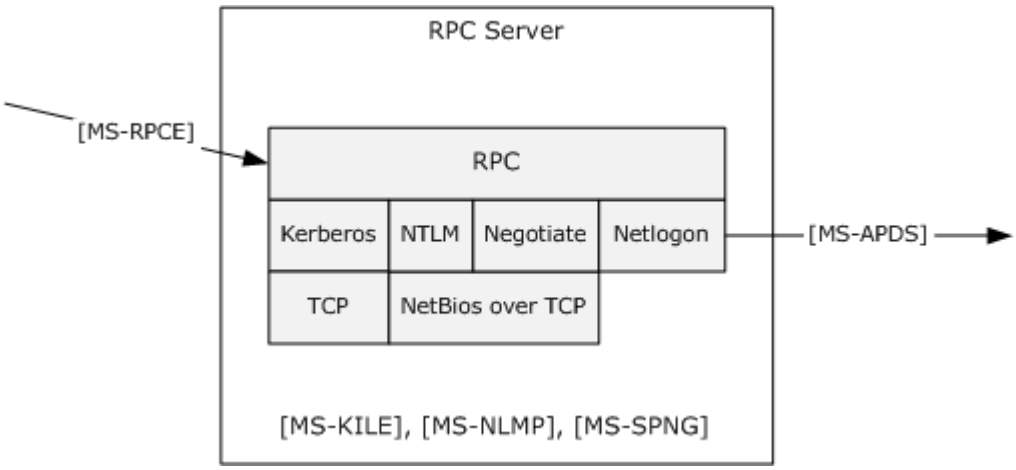

Figure 36: Components of the RPC Server

RPCE supports a wide range of underlying transports including both connection-oriented and connectionless. For connection-oriented transports, RPCE supports transports including direct TCP, SMB [MS-SMB] and [MS-SMB2], NetBIOS over IPX, NetBIOS Extended User Interface (NetBEUI), NetBIOS over TCP, and RPC over HTTP [MS-RPCH].

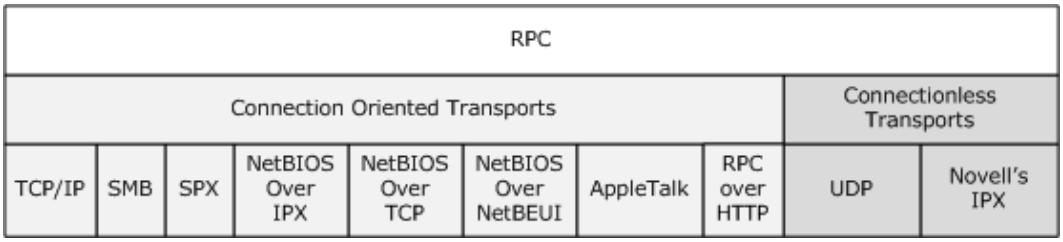

Figure 37: RPC transports

7.2.2 Task Relationships

7.2.2.1 Black Box Relationship Diagrams

The task of Remote Procedure Services - RPC Server uses a number of protocols (some are prerequisite and others are used within the task itself). The RPC client always triggers the task - specifically, the client directs the RPC client to the RPC server beginning the server task. The task involves a server member computer in the domain as well as the authority for the domain, that is, the domain controller. The RPC client may be either a domain-joined computer or a non-domain joined computer.

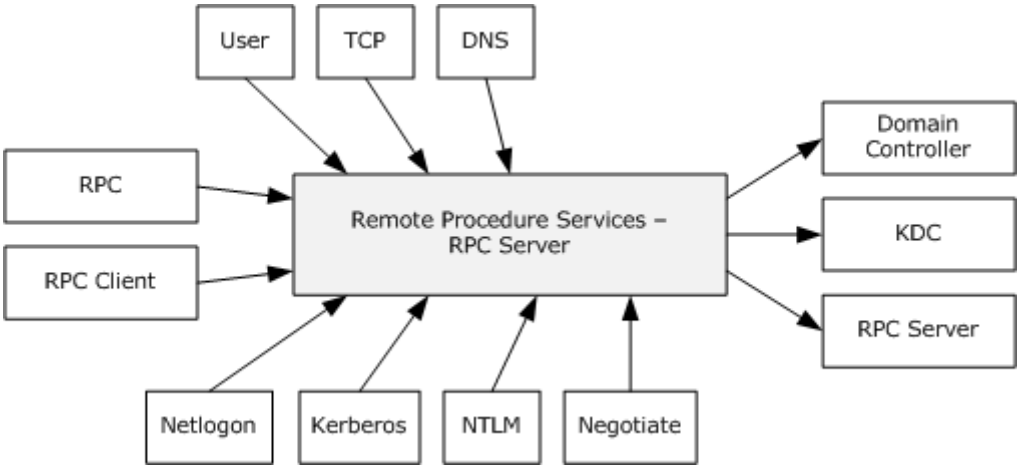

Figure 38: Black box relationships for the Remote Procedure Services Authentication - RPC Server task

7.2.2.2 Task Dependencies

This task has a number of external dependencies. The following protocols are used by this task:

- Precondition - System Level
  - Domain Interaction System Overview [MS-DISO]

- Precondition - Protocols
  - TCP/IP
  - DNS
  - RPC
  - SMB (either SMB or SMB Version 2)
  - Negotiate
  - NTLM
  - Kerberos
  - Netlogon
  - Group Policy
- Precondition - Tasks
  - Server Domain Join
- External Server System
  - Active Directory
  - DNS Directory
  - KDC

7.2.2.3 Task Influences

The task has several external influences. These influences can alter state of the protocols, for example limiting which authentication mechanisms are offered.

External Influences:

- Group Policy for the computer
- Group Policy for the user account for the domain logon
- Active Directory - account lockout subsystem, account restrictions
- Security event logging on the domain controller
- RPC resource access requirements on the procedures executed (authorization)

7.2.3 Task Assumptions and Preconditions

This task has the precondition that the domain controller has been configured to support the domain infrastructure. The user account for the client authentication has been created and provisioned on the domain controller.

Also, as part of the Windows startup procedure for the server computer, the following actions have taken place (see [MS-DISO] for additional details):

- Successful connection to the network and startup of networking stacks

- Successful domain join
- Establishment of a secure channel with the domain controller
- Startup and configuration of the RPC service on both the client and server computers

7.2.4 Task Versioning and Capability Negotiation

RPC supports selection of authentication security providers and negotiation for authentication. This task uses the versioning and negotiation support provided by the respective protocols used for this task. See each Protocol Technical Document for details on a specific protocol used. This task does not have specific support for Task Versioning and capability negotiation beyond the TDs referenced.

7.3 Task Architecture

The basic architecture contains a domain controller, a member server computer, and a client computer. The RPC client is initiating an RPC authentication for the server procedure execution.

7.3.1 Task Architectural Constraints

The primary constraint is that the server computer has successfully accomplished a domain join and is operating within the network boundaries of the domain. The member machines MUST have network communication access to the domain controller. The RPC client MUST have network access to the RPC server.

7.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by the processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

ServerSecurityContext - A security context that represents the server system.

7.3.3 Task Abstract Parameters

This section describes data that is passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements that are presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

LogonServer - The domain controller to be used for authentication

DNSServer - The domain controller to be used for DNS queries

7.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations may depart from this abstraction so long as their external behavior remains consistent with that described in this document.

UserLogonToken - The logon token created during user network logon, derived from the authentication protocol.

7.3.5 White-Box Relationships

The following figure represents the white-box relationships of the task within the client computer, server computer and with the domain controller.

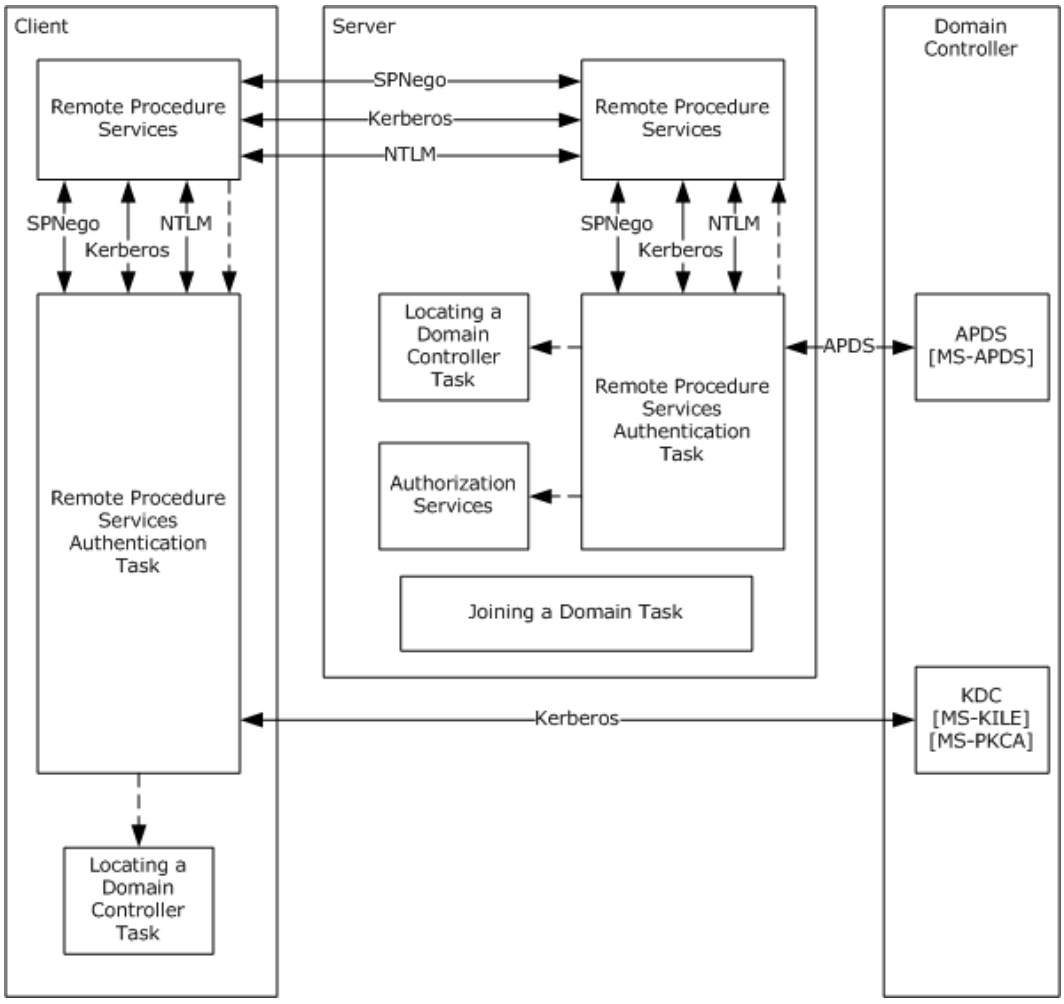

Figure 39: White box relationships for the Remote Procedure Services Authentication - RPC Server task

7.3.6 Task Events

This task initiates audit events based on the success/failure of the task. The authentication audit events are logged at the domain controller for the verification of the user credentials. Authentication audit events and access check events are also logged at the server computer.

7.3.6.1 Task Timers

There are no timers employed directly by this task. Timers may be employed by invoked protocols, such as KILE. Invoked protocol timers are detailed in their respective Protocol Technical Document.

7.3.6.2 Task Non-Timer Events

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

7.3.7 Task Architecture and Communication

The process of building a security context will start on the client. The client triggers the task by initiating the RPC request.

Upon receiving and processing an authentication token at any point in the authentication process on either the client or server, the security provider will indicate to the RPC runtime one of three abstract results from the processing: an error, a success, or a request for further security legs, as specified in [RFC2743]. If the security provider indicates an error, the RPC runtime takes recovery action that is dependent on the location of the error.

In RPCE, the server registers the permitted forms of security providers as listed in section 2.2.1.1.7. The client in initiating the RPC request MUST specify the security provider in the Bind request. Details on the protocol exchange between RPC client and RPC server for connectionless transports can be found in [MS-RPCE] section 3.2.1.4.1. Details for connection-oriented transports can be found in [MS-RPCE] section 3.3.1.5.2.1.

The diagram below details the RPC protocol traffic between the RPC client, RPC server, and domain controller when the RPC client requested secure, connection-oriented RPC using Kerberos security provider.

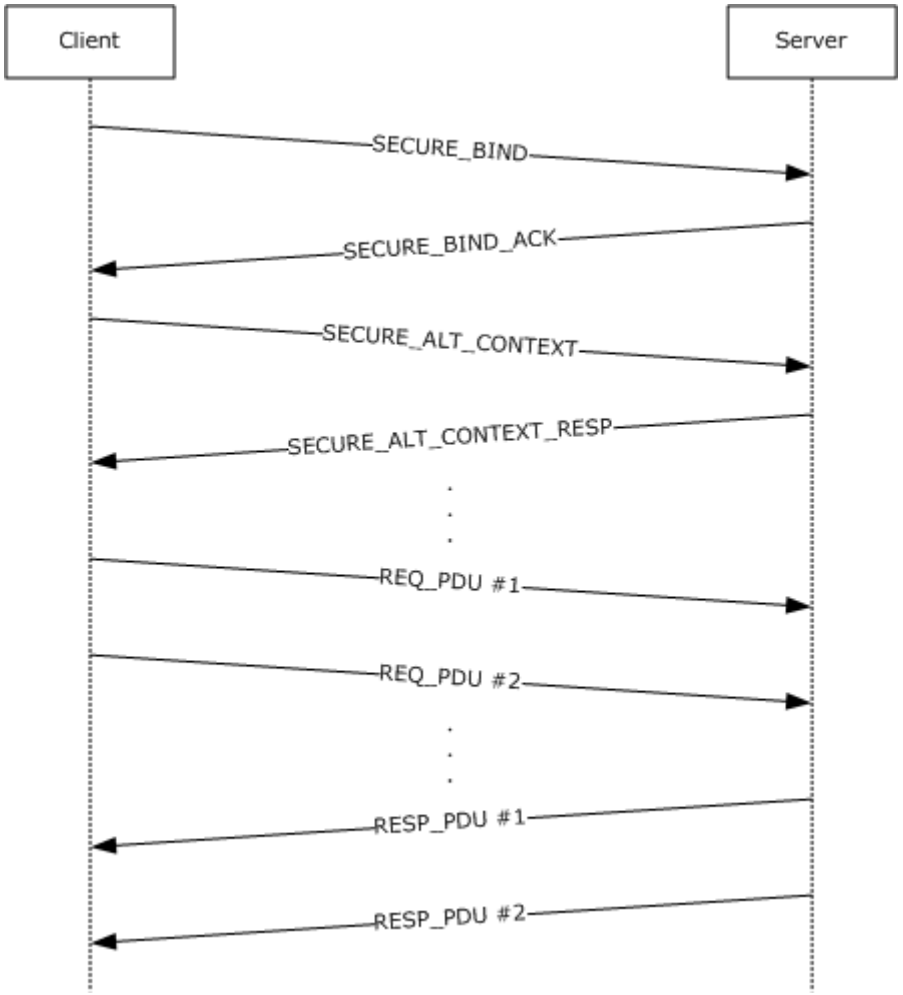

Figure 40: Sequence diagram for RPC using Kerberos as the security provider

To validate a Kerberos ticket in the RPC client's alter_context request, the RPC server uses its Kerberos key.

As a second example, RPCE with secure, connection-oriented RPC with NTLM as the security provider can be illustrated as follows.

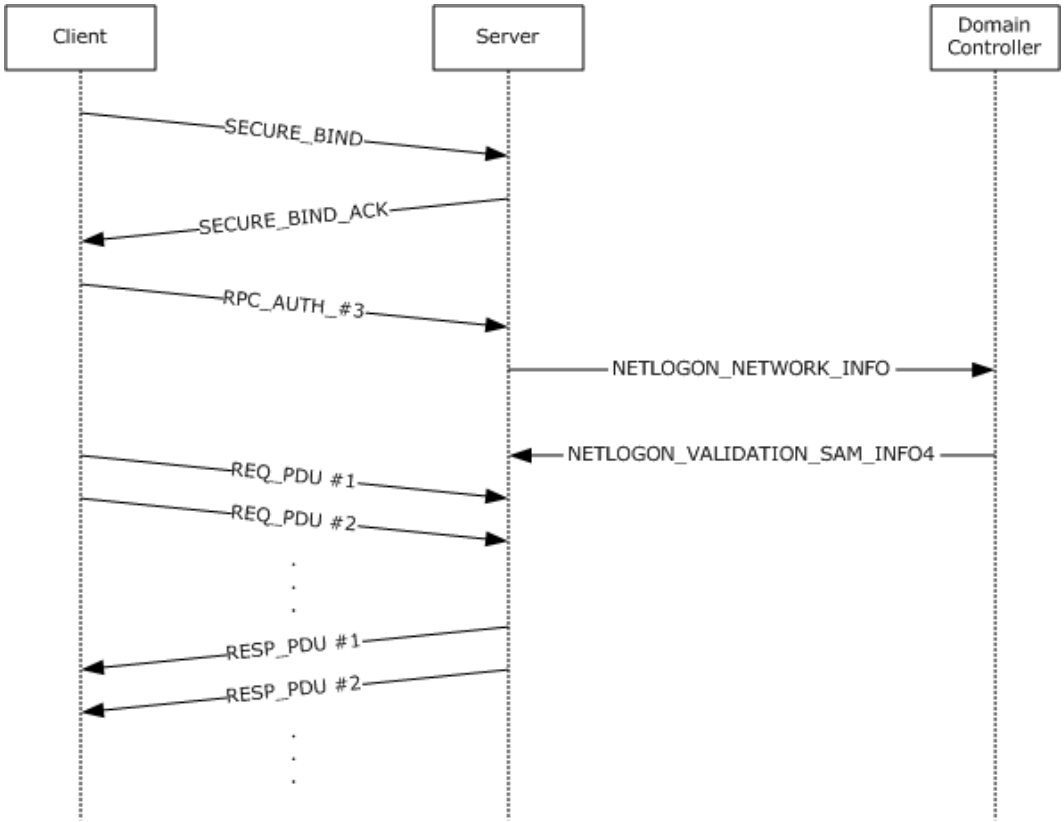

Figure 41: Sequence diagram for RPC with NTLM as the security provider

In the final leg of the NTLM exchange (the third leg), the authentication material in RPC_AUTH_3 PDU is extracted. Upon receiving this PDU, the server calls the implementation equivalent of the abstract GSS_Accept_sec_context for NLMP, which communicates with the domain controller via APDS to validate the user authentication. In the preceding example, the APDS response returns success and NLMP validation succeeds.

7.3.8 Task Processing Rules

Init RPCAccessAuth:

Step 1: Check to see if the server computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

Step 3: The server computer starts up RPC service and registers configuration, including supported security providers.

Step 4: The user provides credential information to the RPC client for authenticating RPC requests to the remote RPC server.

Step 5: The client computer sends an RPC request to the server.

Step 6: The server is waiting for client requests, and the server receives an RPC request from the client and passes it to the RPC service.

Task RPCAccessAuth:

Step 1: The RPC server evaluates the authentication parameters in the RPC request.

Step 2: If the client provided unacceptable authentication parameters, the server returns a Fault message, closes the connection, and goes to "Init RPCAccessAuth:Step 6", shown above, where it waits for client requests.

Step 3: The server produces a response containing authentication data for the client. The authentication data is generated by a server-supported security provider.

Step 4: The client and server continue to exchange RPC messages until server authentication protocols determine that a maximum number of exchanges has taken place or the SMB server has sufficient information to attempt to authenticate the connection requests.

Step 5: The server security provider contacts the domain controller to validate the authentication information.

Step 6: If the domain controller validates the authentication information, the RPC server returns a successful authentication response message; otherwise, the RPC server returns a failure message to the RPC client.

Each of the primary steps within this task can return errors. Appropriate error handling SHOULD be done.

7.3.9 Task Failure Scenarios

Each of the primary steps in the Remote Procedure Services Authentication - RPC Server task can result in a failure case. On any error not directly specified in the processing tasks above, the task returns an RPC Fault message and fails the RPC request and closes the connection.

Some of the protocols contain recoverable errors that are handled in the lower-level protocols. For example, in the MS-KILE protocol, the KDC will be contacted during Step 5. The MS-KILE protocol will retry contacting the KDC up to three times if there is no response. This failure/recovery path is handled at a lower protocol level and is not directly influenced by the task.

7.4 Task Details

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

7.4.1 Task Precondition Details

This task has the precondition that the domain controller has been configured to support the domain infrastructure. The user account for the client network logon has been created and provisioned on the domain controller. Networking support has been established between the client, server, and domain controller.

Also, as part of the Windows startup procedure for the server computer, the actions specified in section 4.4.1 have taken place (see [MS-DISO] for additional details). The server computer has successfully joined a domain.

Step 1: Check to see if the server computer has successfully joined a domain.

Step 2: If not joined to a domain, return error Not_Domain_Joined.

The following additional steps will be undertaken:

Step 3: The server computer starts up the RPC service. The server is required to run a service to support RPC protocol server-side operations as specified in [MS-RPCE]. The RPC server registers the configuration of the RPC service including the acceptable authentication security providers. A list of the acceptable security providers is provided in [MS-RPCE] section 2.2.1.1.7.

Step 4: The user provides credential information to the RPC client for authenticating RPC requests to the remote RPC server.

Step 5: The client computer sends an RPC request to the server. The client computer initiates an RPC request conforming to RPCE on behalf of the user. As part of the secure RPC request, the RPC client initiates the secure bind beginning the RPC message exchange by specifying the security provider used for the secure bind and providing the security token output from the GSS Init_security_context call. Details on this action are provided in [MS-RPCE] section 2.2.2.12. The acceptable security providers are listed in [MS-RPCE] section 2.2.1.1.7.

Step 6: The server is waiting for client requests, and the server receives an RPC request from the RPC client and passes it to the RPC service.

7.4.2 Task Initialization of External Entities

This task does not initialize external entities. The external protocols used by this task have been initialized during system startup and during the domain join task.

7.4.3 Task Event Details

7.4.3.1 Task Timer Details

There are no timers employed directly by this task. Timers may be employed by invoked protocols, such as KILE. Invoked protocol timers are detailed in their respective Protocol Technical Document.

7.4.3.2 Task Non-Timer Event Details

There are no non-timer events employed directly by this task. Non-timer events may be employed by other protocols or tasks after this task has completed. Such non-timer events are detailed in their respective Defined Task System Document or Protocol Technical Document.

7.4.4 Task Architectural Details

The following figure illustrates the steps that make up the task in a flow diagram.

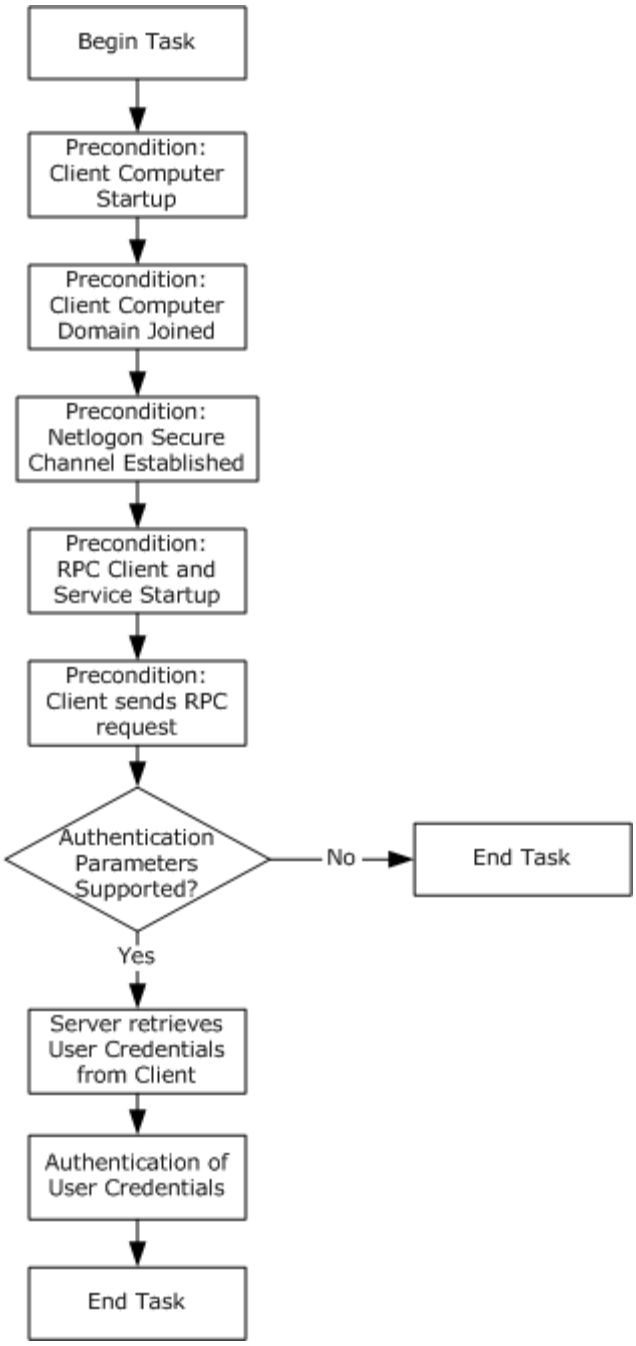

Figure 42: Flow diagram for the Remote Procedure Services Authentication - RPC Server task

The following figure illustrates the high level components that make up the task. The protocols have been initialized during member machine system startup and during domain join; specifically, Kerberos has detected the KDC for the domain, and networking connections are established between the client, server, and domain controller.

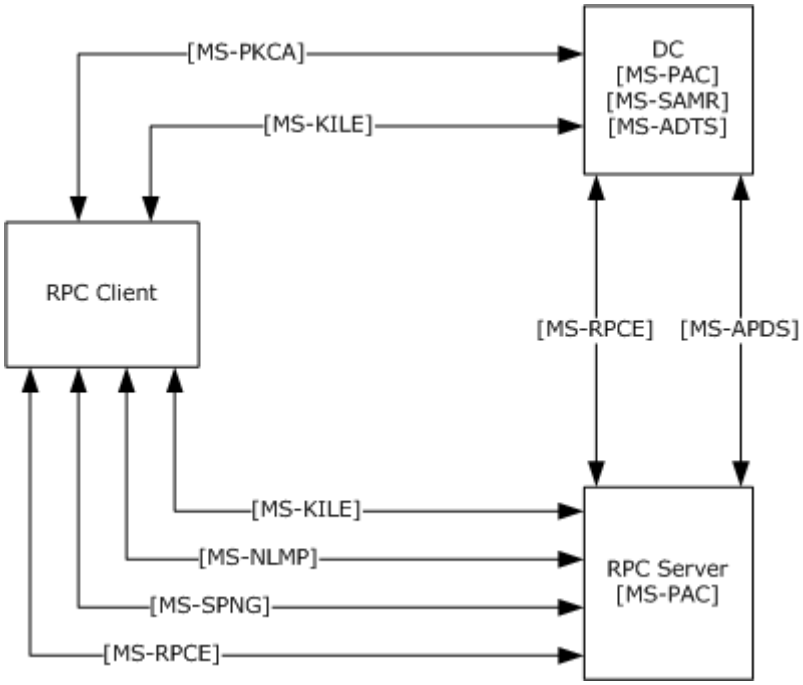

Figure 43: Standards used by the Remote Procedure Services Authentication - RPC Server task

The RPC client and server can use one of several authentication mechanisms as detailed in [MS-RPCE]. The RPC server registers the acceptable security authentication protocols. The RPCE protocol can use NTLM or Negotiate, which may select from Kerberos or NTLM. If Kerberos is negotiated, the KILE protocol will be used by the client to contact the domain controller to obtain a ticket. The client provides the ticket to the server via KILE encapsulated within the Negotiate Protocol with the RPC request.

The authentication token information is contained within the RPC request and response according to the authentication standards and TDs. [MS-RPCE] sections 2.2.2.12, 3.2.1.4.1, and 3.3.1.5.2 detail the process to obtain the authentication tokens and their use in the RPC messages.

The server, depending on the authentication mechanism used in the RPC request, will use [MS-APDS] section 1.4, as described in the Technical Document.

7.4.5 Task Processing Rule Details

After the server's system startup and domain join have been completed as summarized in section 7.4.1, the RPC server can process the RPC authentication requests.

The detailed sequence is as follows:

Step 1: The RPC server evaluates the authentication parameters in the RPC request.

Step 2: If the client provided unacceptable authentication parameters, the server returns a Fault message, closes the connection, and goes to Step 6 in section 7.4.1, where it waits for client requests.

Step 3: The server produces a response containing authentication data for the client. The authentication data is generated by a server-supported security provider.

Step 4: The client and server continue to exchange RPC messages until the server authentication protocols determine that a maximum number of exchanges has taken place or the SMB server has sufficient information to attempt to authenticate the connection requests.

Step 5: The server authentication provider contacts the domain controller to validate the authentication information.

Step 6: If the domain controller validates the authentication information, the RPC server returns a successful authentication response message; otherwise, the RPC server returns a failure message to the RPC client. On success, a logon token, UserLogonToken, is created within the security context, ServerSecurityContext, on the server.

Within these protocols on the server, the APDS protocol is used to communicate with the domain controller. For each of these protocols, APDS inputs are extracted from the SecurityBlob (auth_token) contained in the alter_context PDUs request.

7.5 Task Security

This section documents security issues specific to this task that are not otherwise described in the Technical Documents (TDs) for the protocols used in the task. It does not duplicate what is already in the protocol TDs unless there is some unique aspect that applies to the system as a whole.

Microsoft RPC provides multiple levels of services based on authentication. Depending on the authentication level, the origin of the traffic (which security principal sent the traffic) can be verified when the connection is established, when the client starts a new remote procedure call, or during each packet exchange between the client and server.

Even when the sender of the traffic is verified, security is still weak, since such verification does not ensure that the packet was not modified or corrupted en route; it only verifies that the packet came from the given principal. For greater security, distributed applications can configure RPC to verify that none of the data exchanged between the client and server is modified. RPC can also encrypt the contents of every packet before sending it. In general, applications that want to secure their traffic will use only the last two levels: integrity and privacy.

The task security considerations are described in their respective Technical Documents.

8 Security

This section documents security issues common to all tasks that are not otherwise described in the Technical Documents (TDs) for the protocols used in the task. It does not duplicate what is already in the protocol TDs unless there is some unique aspect that applies to the system as a whole.

Security considerations are provided under each task's section on Task Security or within the Technical Documents of the protocols used for each task.

After authentication is finalized, a security context is maintained by both client and server implementations for the lifetime of the activity the security context has been established for, unless the security provider indicates that the context has expired.

Be aware that higher levels of authentication and subsequent message protection require higher computational overhead. The client and the server portions of the distributed application MUST use the same authentication level.

9 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Server 2003 R2 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

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

<1> Section 5.3.2: Negotiate and Kerberos are not offered in Windows NT.

<2> Section 5.4.5: "Nego2" mapping applies to Windows 7 and Windows Server 2008 R2.

10 Change Tracking

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

11 Index