[MS-APDS]:

Authentication Protocol Domain Support

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
2/22/2007	0.1	New	Version 0.1 release
6/1/2007	2.0	Major	Updated and revised the technical content.
7/3/2007	3.0	Major	Added new protocol.
7/20/2007	3.0.1	Editorial	Changed language and formatting in the technical content.
8/10/2007	3.0.2	Editorial	Changed language and formatting in the technical content.
9/28/2007	4.0	Major	Updated and revised the technical content.
10/23/2007	4.0.1	Editorial	Changed language and formatting in the technical content.
11/30/2007	4.0.2	Editorial	Changed language and formatting in the technical content.
1/25/2008	4.0.3	Editorial	Changed language and formatting in the technical content.
3/14/2008	5.0	Major	Updated and revised the technical content.
5/16/2008	5.0.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	6.0	Major	Updated and revised the technical content.
7/25/2008	7.0	Major	Updated and revised the technical content.
8/29/2008	7.0.1	Editorial	Changed language and formatting in the technical content.
10/24/2008	7.0.2	Editorial	Changed language and formatting in the technical content.
12/5/2008	8.0	Major	Updated and revised the technical content.
1/16/2009	9.0	Major	Updated and revised the technical content.
2/27/2009	9.1	Minor	Clarified the meaning of the technical content.
4/10/2009	10.0	Major	Updated and revised the technical content.
5/22/2009	11.0	Major	Updated and revised the technical content.
7/2/2009	12.0	Major	Updated and revised the technical content.
8/14/2009	13.0	Major	Updated and revised the technical content.
9/25/2009	14.0	Major	Updated and revised the technical content.
11/6/2009	15.0	Major	Updated and revised the technical content.
12/18/2009	16.0	Major	Updated and revised the technical content.
1/29/2010	17.0	Major	Updated and revised the technical content.
3/12/2010	17.0.1	Editorial	Changed language and formatting in the technical content.
4/23/2010	18.0	Major	Updated and revised the technical content.
6/4/2010	19.0	Major	Updated and revised the technical content.
7/16/2010	20.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
8/27/2010	20.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	21.0	Major	Updated and revised the technical content.
11/19/2010	21.1	Minor	Clarified the meaning of the technical content.
1/7/2011	21.1	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	22.0	Major	Updated and revised the technical content.
3/25/2011	22.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	23.0	Major	Updated and revised the technical content.
6/17/2011	23.1	Minor	Clarified the meaning of the technical content.
9/23/2011	23.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	24.0	Major	Updated and revised the technical content.
3/30/2012	24.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	24.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	25.0	Major	Updated and revised the technical content.
1/31/2013	25.1	Minor	Clarified the meaning of the technical content.
8/8/2013	26.0	Major	Updated and revised the technical content.
11/14/2013	26.1	Minor	Clarified the meaning of the technical content.
2/13/2014	27.0	Major	Updated and revised the technical content.
5/15/2014	28.0	Major	Updated and revised the technical content.
6/30/2015	29.0	Major	Significantly changed the technical content.
10/16/2015	30.0	Major	Significantly changed the technical content.
7/14/2016	31.0	Major	Significantly changed the technical content.
6/1/2017	31.0	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	32.0	Major	Significantly changed the technical content.
9/12/2018	33.0	Major	Significantly changed the technical content.
4/7/2021	34.0	Major	Significantly changed the technical content.
6/25/2021	35.0	Major	Significantly changed the technical content.
4/9/2024	36.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

Authentication Protocol Domain Support (APDS) specifies the required communication between a server and a domain controller (DC) that uses Netlogon interfaces to complete an authentication sequence.

An operating system can support several authentication protocols, such as NT LAN Manager (NTLM) Authentication Protocol, Kerberos, Secure Sockets Layer (SSL)/Transport Layer Security (TLS), and Digest authentication. APDS is used by NT LAN Manager (NTLM) and the Digest validation protocol to validate the user's credentials at the domain controller. The Kerberos protocol uses APDS to perform the required communication for privilege attribute certificate (PAC) validation.

All these protocols can be supported by any server and rely only on a local user account database except for Kerberos that relies on a mutually trusted third-party called Key Distribution Center (KDC) [MS-KILE]. Therefore, specifications for these protocols can stand entirely on their own. However, in a domain context, when the server is a member of a domain and relies on the domain account database, the domain controller contributes to the authentication and authorization processes. Domain members use the Netlogon Remote Protocol [MS-NRPC] to communicate with the domain controller for purposes of authentication and authorization.

The implementations of these authentication protocols use a variety of methods to communicate with the domain controller in the course of their executions. These methods, collectively referred to as Authentication Protocol Domain Support, are specified in this document.

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

1.1 Glossary

This document uses the following terms:

Active Directory: The Windows implementation of a general-purpose directory service, which uses LDAP as its primary access protocol. **Active Directory** stores information about a variety of objects in the network such as user accounts, computer accounts, groups, and all related credential information used by **Kerberos** [MS-KILE]. **Active Directory** is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS), which are both described in [MS-ADOD]: Active Directory Protocols Overview.

APDS server: The server side of Authentication Protocol Domain Support [MS-APDS], otherwise known as a **domain controller** in authentication protocols that use Authentication Protocol Domain Support.

application server: The server side of Kerberos Network Authentication Service (V5) Extensions [MS-KILE].

Digest authentication: A protocol that uses a challenge-response mechanism for authentication in which clients are able to verify their identities without sending an in-the-clear password to the server. For more information, see [RFC2617] and [RFC2831].

Digest client: The Digest Access Authentication: Microsoft Extensions [MS-DPSP] client.

Digest server: The server side of Digest Access Authentication: Microsoft Extensions [MS-DPSP].

Digest validation: A protocol to verify the **Digest authentication** challenge-response from a client to a server for a specified **domain** account.

directory: The database that stores information about objects such as users, groups, computers, printers, and the directory service that makes this information available to users and applications.

domain: A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set has to act as a **domain controller (DC)** and host a member list that identifies all members of the domain, as well as optionally hosting the **Active Directory** service. The domain controller provides authentication of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [MS-AUTHSOD] section 1.1.1.5 and [MS-ADTS].

domain account: A stored set of attributes representing a principal used to authenticate a user or machine to an **Active Directory** domain.

domain controller (DC): The service, running on a server, that implements **Active Directory**, or the server hosting this service. The service hosts the data store for objects and interoperates with other **DCs** to ensure that a local change to an object replicates correctly across all **DCs**. When **Active Directory** is operating as Active Directory Domain Services (AD DS), the **DC** contains full NC replicas of the configuration naming context (config NC), schema naming context (schema NC), and one of the domain NCs in its forest. If the AD DS **DC** is a global catalog server (GC server), it contains partial NC replicas of the remaining domain NCs in its forest. For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. When **Active Directory** is operating as Active Directory Lightweight Directory Services (AD LDS), several AD LDS **DCs** can run on one server. When **Active Directory** is operating as AD DS, only one AD DS **DC** can run on one server. However, several AD LDS **DCs** can coexist with one AD DS **DC** on one server. The AD LDS **DC** contains full NC replicas of the config NC and the schema NC in its forest. The domain controller is the server side of Authentication Protocol Domain Support [MS-APDS].

interactive logon: A software method in which the account information and credentials input by the user interactively are authenticated by a server or **domain controller (DC)**.

Kerberos: An authentication system that enables two parties to exchange private information across an otherwise open network by assigning a unique key (called a ticket) to each user that logs on to the network and then embedding these tickets into messages sent by the users. For more information, see [MS-KILE].

Key Distribution Center (KDC): The **Kerberos** service that implements the authentication and ticket granting services specified in the **Kerberos** protocol. The service runs on computers selected by the administrator of the **realm** or domain; it is not present on every machine on the network. It has to have access to an account database for the **realm** that it serves. **KDCs** are integrated into the **domain controller** role. It is a network service that supplies tickets to clients for use in authenticating to services.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

machine account: An account that is associated with individual client or server machines in an **Active Directory** domain.

NetBIOS: A particular network transport that is part of the LAN Manager protocol suite. **NetBIOS** uses a broadcast communication style that was applicable to early segmented local area networks. A protocol family including name resolution, datagram, and connection services. For more information, see [RFC1001] and [RFC1002].

network logon: A software method in which the account information and credentials previously supplied by the user as part of an interactive logon are used again to log the user onto another network resource.

NT LAN Manager (NTLM) Authentication Protocol: A protocol using a challenge-response mechanism for authentication in which clients are able to verify their identities without sending a password to the server. It consists of three messages, commonly referred to as Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication).

NTLM client: The NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP] client.

NTLM server: The server side of NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP].

NTOWF: In the context of an NTLM authentication, a NT LAN Manager (NT) one-way function (OWF) used to create a hash based on the user's password to generate a principal's secret key. The NTLM hash superseded the LAN Manager (LM) hash.

principal: A unique entity identifiable by a security identifier (SID) that is typically the requester of access to securable objects or resources. It often corresponds to a human user but can also be a computer or service. It is sometimes referred to as a security principal.

privilege attribute certificate (PAC): A Microsoft-specific authorization data present in the authorization data field of a ticket. The **PAC** contains several logical components, including group membership data for authorization, alternate credentials for non-Kerberos authentication protocols, and policy control information for supporting interactive logon.

realm: An administrative boundary that uses one set of authentication servers to manage and deploy a single set of unique identifiers. A realm is a unique logon space.

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

RPC transport: The underlying network services used by the remote procedure call (RPC) runtime for communications between network nodes. For more information, see [C706] section 2.

Secure Sockets Layer (SSL): A security protocol that supports confidentiality and integrity of messages in client and server applications that communicate over open networks. SSL supports server and, optionally, client authentication using X.509 certificates [X509] and [RFC5280]. SSL is superseded by **Transport Layer Security (TLS)**. TLS version 1.0 is based on SSL version 3.0 [SSL3].

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

service: A process or agent that is available on the network, offering resources or services for clients. Examples of services include file servers, web servers, and so on.

Transport Layer Security (TLS): A security protocol that supports confidentiality and integrity of messages in client and server applications communicating over open networks. TLS supports server and, optionally, client authentication by using X.509 certificates (as specified in [X509]). TLS is standardized in the IETF TLS working group.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

user principal name (UPN): A user account name (sometimes referred to as the user logon name) and a domain name that identifies the domain in which the user account is located. This is the standard usage for logging on to a Windows domain. The format is: someone@example.com (in the form of an email address). In **Active Directory**, the userPrincipalName attribute of the account object, as described in [MS-ADTS].

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

1.2 References

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

1.2.1 Normative References

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.

[ISO/IEC-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=28245

Note There is a charge to download the specification.

[MS-ADA1] Microsoft Corporation, "Active Directory Schema Attributes A-L".

[MS-ADA2] Microsoft Corporation, "Active Directory Schema Attributes M".

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

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

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

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

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

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

[MS-LSAD] Microsoft Corporation, "Local Security Authority (Domain Policy) Remote 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-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

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

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

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

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

[RFC4757] Jaganathan, K., Zhu, L., and Brezak, J., "The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows", RFC 4757, December 2006, https://www.rfc-editor.org/info/rfc4757

1.2.2 Informative References

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

[MSDN-0SUBAUTHROUTINE9] Microsoft Corporation, "Msv1_0SubAuthenticationRoutine function (subauth.h)", 9 Parameters, Security, https://learn.microsoft.com/en-us/windows/win32/api/subauth/nf-subauth-msv1_0subauthenticationroutine

[RFC1994] Simpson, W, "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996, https://www.rfc-editor.org/info/rfc1994

[RFC2069] Franks, J., et al., "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997, http://www.ietf.org/rfc/rfc2069.txt

[X509] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks", Recommendation X.509, August 2005, http://www.itu.int/rec/T-REC-X.509/en

1.3 Overview

Authentication protocols such as NT LAN Manager (NTLM), **Kerberos**, **Secure Sockets Layer (SSL)**/**Transport Layer Security (TLS)**, and **Digest authentication** are used by a variety of higher-layer protocols to provide security **services**.

The Authentication Protocol Domain Support Protocol specifies the communication between the server and the **domain controller** for each of the protocols.

Each of the protocols has a specific exchange with the domain controller (DC) as follows:

- Authenticate the client: **NTLM** and digest.
- Obtain authorization information, such as group memberships: NTLM, digest, and SSL/TLS.
- Verify the authorization information: The server operating system for Kerberos **privilege attribute certificate (PAC)** [MS-PAC].

All these back-end, server-to-server protocols in turn use the Netlogon Remote Protocol [MS-NRPC] for their transport to the DC. Specifically, the protocols behave as follows:

- The NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP] uses the Netlogon Remote Protocol [MS-NRPC] to communicate with the DC to complete the authentication of a **domain account** during an **interactive logon** or **network logon**. As user account information is maintained by the DC, only the DC can validate user credentials and complete the authentication sequence. The server then uses the authorization information returned by the DC to make authorization decisions.
- The server operating system uses Netlogon generic pass-through ([MS-NRPC] section 3.2) to validate the PAC that it receives in the ticket from the client. Because PAC information can be altered by the server, the operating system might contact the DC to validate the PAC and ensure its integrity.
- The Digest Protocol Extensions [MS-DPSP] is used by deployments in which users are authenticated based on user name and password by using the Digest authentication mechanism. The Digest authentication mechanism itself defines how the client authenticates the user to the server (by proving knowledge of the password), and optionally provides integrity and confidentiality of subsequent messages exchanged between the client and the server. **Digest validation** is performed between the server and the DC during the initial client/server Digest–based authentication as follows:

1. The server which does not have access to the user's password sends a Digest validation request message (section 2.2.5.1) to the domain controller by using the generic pass-through capability of the Netlogon Remote Protocol.

2. The DC looks up the user's password and uses it to verify the validity of the digest input. The digest input is originally generated by the **Digest client** using the user's password, as specified in [RFC2617] and [RFC2831]).

3. On successful validation, the domain controller returns the PAC in the DIGEST_VALIDATION_RESP message. The PAC represents the user's identity and group memberships, suitable for making authorization decisions.

- SSL/TLS and other protocols that authenticate users via the X.509 certificates [X509] can use the Remote Certificate Mapping Protocol [MS-RCMP], which relies on the generic pass-through capability of Netlogon to retrieve authorization information associated with users.

1.4 Relationship to Other Protocols

Each of the following protocols relies on the Netlogon Remote Protocol ([MS-NRPC]) to complete the authentication sequence and retrieve or validate authorization information associated with a user. All higher-layer protocols that use one of the protocols specified in this document (for example, **NT LAN Manager (NTLM) Authentication Protocol** [MS-NLMP]) leverage the functionality specified here when in a **domain** environment.

1.4.1 NTLM Logon

NTLM authentication ([MS-NLMP]) uses the Netlogon pass-through authentication ([MS-NRPC] section 3.2) to authenticate the user in the **domain** with the **domain controller** during an **interactive logon** or a **network logon**.

1.4.2 Kerberos PAC Validation

The server operating system uses the Netlogon generic pass-through ([MS-NRPC] section 3.2.4.1) to validate the **privilege attribute certificate (PAC)** with the **domain controller**, when required.

1.4.3 Digest Validation Protocol

The **Digest validation** defined in this document relies on the generic pass-through capability of the Netlogon Remote Protocol ([MS-NRPC] section 3.2.4.1) as a transport for the DIGEST_VALIDATION_REQ (section 2.2.5.1) and DIGEST_VALIDATION_RESP (section 2.2.5.2) messages.

1.5 Prerequisites/Preconditions

Each of the protocols in this specification relies on the Netlogon Remote Protocol [MS-NRPC]. [MS-NRPC] assumes that the following prerequisites are met:

- The server has a **machine account** in the **domain**.
- The server has established a secure connection with the **domain controller (DC)**.

1.5.1 NTLM Logon

NTLM interactive logon and NTLM **network logon** have all of the prerequisites and preconditions that are defined in the NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP].

1.5.2 Kerberos PAC Validation

Kerberos PAC validation assumes that the server operating system has a **PAC** that has been received in a ticket from the client to the **application server**. It is possible for a server operating system to require PAC validation (section 1.6.2).

1.5.3 Digest Validation Protocol

The Digest validation protocol assumes the following:

- The **DC** server has access to the user's password.
- The **Digest client** possesses the digest-response message ([RFC2617] section 3.2.2) and the initial digest-challenge message ([RFC2617] section 3.2.1) used in the **Digest authentication** protocol.

1.6 Applicability Statement

All protocol support for **domains** requires a domain authority to process the requests. These protocols are not applicable to any stand-alone machine that is not associated with a domain. Each protocol has additional applicability constraints.

1.6.1 NTLM Logon

NTLM ([MS-NLMP] section 1.6) also applies when the **NTLM server** performing the NTLM authentication is a member of a **domain**. Both NTLM for **interactive logon** and NTLM for **network logon** can be performed by using a **domain controller (DC)**.

Network logon using NTLM is applicable under any one of the following situations:

- Client or application is explicitly configured to use NTLM.
- Client or application uses an invalid target name (the server's **principal** name) and cannot log on by using **Kerberos**.

Client or application specifies a target name that cannot be resolved and, therefore, cannot log on by using Kerberos.

1.6.2 Kerberos PAC Validation

Figure 1: Kerberos PAC validation

Before Kerberos PAC validation occurs, the client has sent the **privilege attribute certificate (PAC)** to the **service** as a part of the Kerberos Protocol Extensions described in [MS-KILE]. The operating system on which the service runs validates the PAC to prevent PAC tampering by the service. PAC tampering can result in inappropriate elevation of privileges.

PAC validation is applicable for **Kerberos** applications that process and interpret the PAC and present that authorization data to additional services. It is optional for a self-contained application because the security threat that the protocol addresses is not relevant for self-contained applications.

1.6.3 Digest Validation Protocol

The Digest validation protocol is appropriate for servers that are implementing **Digest authentication** and are acting as members in an **Active Directory**–compatible **domain**.

1.7 Versioning and Capability Negotiation

1.7.1 NTLM Logon

NTLM interactive logon and **network logon** do not have any versioning or capability negotiation.

1.7.2 Kerberos PAC Validation

Kerberos PAC validation does not have any versioning or capability negotiation.

1.7.3 Digest Validation Protocol

The DIGEST_VALIDATION_REQ and DIGEST_VALIDATION_RESP messages have a dedicated version number field. This document defines version 1 of the **Digest validation** protocol. The Digest validation protocol does not support any capability negotiation.

1.8 Vendor-Extensible Fields

1.8.1 NTLM Logon

NTLM interactive logon and **network logon** do not have any vendor-extensible fields.

1.8.2 Kerberos PAC Validation

Kerberos PAC validation does not have any vendor-extensible fields.

1.8.3 Digest Validation Protocol

The Digest validation protocol does not have any vendor-extensible fields. Note that the **Digest validation** protocol has reserved fields for future use, but these fields are not intended to carry opaque or vendor-defined data.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Domain support for the Authentication Protocol Domain Support protocol SHOULD use Netlogon **remote procedure call (RPC)** messages in the logon interface. The Netlogon **RPC transport** is specified in [MS-NRPC].

2.2 Message Syntax

For **domain** support, authentication protocols MUST use an NRPC pass-through authentication ([MS-NRPC] section 3.2) method with parameters determined by the authentication protocol being used.

Interactive and **network logon** information, passed through the *LogonInformation* parameter, is used when calling the NetrLogonSamLogonEx method ([MS-NRPC] section 3.5.4.5.1). **Domain controller Kerberos PAC** validation and digest messages MUST be encoded as opaque blobs and transported by the generic pass-through capability of Netlogon ([MS-NRPC] section 3.2.4.1).

All message fields, including bit flags, are in the following sections in **little-endian** format. These data structures MUST be built as if they are on a little-endian machine before transmission. On reception, the messages MUST be interpreted as little-endian and transformed into the native endianness of the implementation.

The following table shows a few of the main status codes returned by these protocols. For a complete list of status codes, see [MS-ERREF].

Symbolic name	Value	Meaning
STATUS_SUCCESS	0x00000000	Requested operation succeeded.
STATUS_LOGON_FAILURE	0xC000006D	Authentication failed.
STATUS_NO_SUCH_USER	0xC0000064	Specified account does not exist.
STATUS_NO_LOGON_SERVERS	0xC000005E	None of the domain controllers are reachable to service the request.

2.2.1 NTLM Logon Message Syntax

The specific message syntax for **NTLM interactive logon** or **network logon** is part of the NRPC pass-through authentication ([MS-NRPC] section 3.2).<1> The **domain** support for NTLM logon is invoked by calling an NRPC pass-through authentication method.

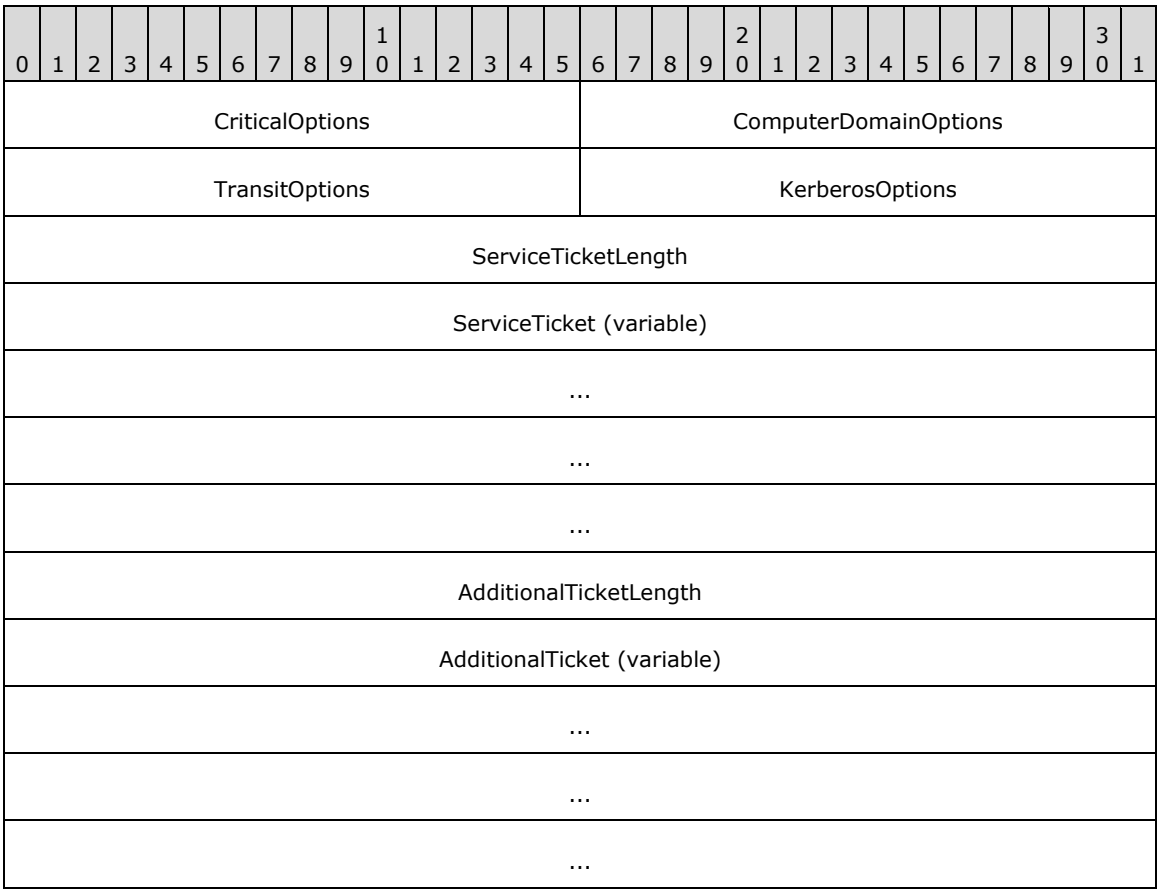
2.2.2 Kerberos Ticket Validation Message Syntax

The Netlogon Ticket Logon Info validation request message, NETLOGON_TICKET_LOGON_INFO (section 2.2.2.1), MUST be encoded as a contiguous buffer. The encoded data SHOULD be sent by using the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1).

2.2.2.1 NETLOGON_TICKET_LOGON_INFO Message

The NETLOGON_TICKET_LOGON_INFO message is used by Kerberos to invoke the network ticket logon flow. In this flow, it calls Netlogon with the ticket which relays the ticket to the issuing domain in

the same fashion as generic passthrough. The NETLOGON_VALIDATION_TICKET_LOGON message (section 2.2.3.1) then processes the validation. This message is defined with the following fields.



CriticalOptions (2 bytes): A USHORT that contains flags that must be understood to parse the rest of the request. The following flag is defined.

Value	Meaning
NoAuthorizationData 0x0000	Only check the ticket; don't return authorization data.

ComputerDomainOptions (2 bytes): A USHORT that contains operations performed by Netlogon in the computer's domain. The following operations are defined.

Value	Meaning
SkipResourceGroups 0x0010	Don't add resource groups from the computer's domain.
SkipA2AChecks 0x0011	Don't perform check A2A and A2ATo access checks.

TransitOptions (2 bytes): A USHORT that contains operations performed by Netlogon at every hop. The following operations are defined.

Value	Meaning
SkipSIDFilter 0x0020	Don't SIDs and transform claims.
SkipNamespaceFilter 0x0021	Don't filter the user domain against the trust's namespace.

KerberosOptions (2 bytes): A USHORT that contains operations performed by the KDC in the ticket's issuing realm. The following operations are defined.

Value	Meaning
SkipPacSignatures 0x0030	Don't verify signatures present in the PAC.
RemoveResourceGroups 0x0031	Strip resource groups from the service ticket.

ServiceTicketLength (4 bytes): A ULONG that contains the length of the preceding service ticket.

ServiceTicket (variable): A pointer to a UCHAR. The Kerberos service ticket that's the source of authorization information.

AdditionalTicketLength (4 bytes): A ULONG that contains the length of the preceding additional ticket.

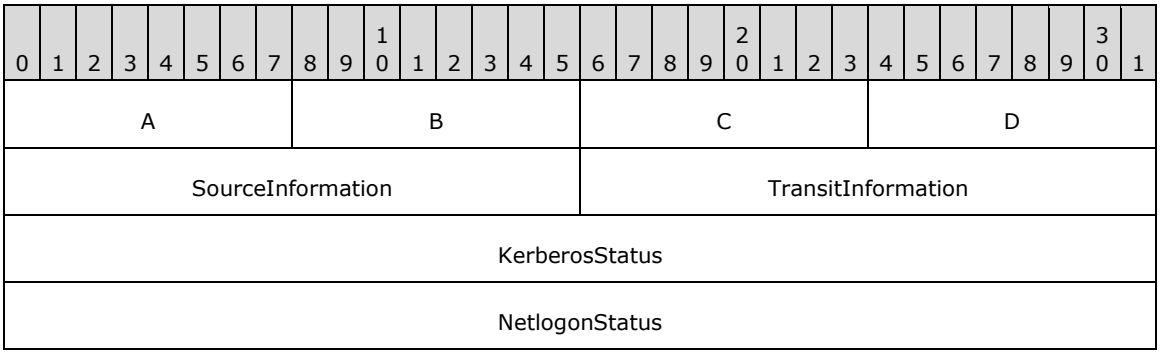
AdditionalTicket (variable): A pointer to a UCHAR. If the service ticket is a User2User ticket then the TGT used as the source of the session key must also be provided.

2.2.3 Kerberos Ticket Validation Response Message Syntax

The Netlogon Ticket Logon Info validation response message, NETLOGON_VALIDATION_TICKET_LOGON (section 2.2.3.1), MUST be encoded as a contiguous buffer. The encoded data SHOULD be sent by using the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1).

2.2.3.1 NETLOGON_VALIDATION_TICKET_LOGON message

The NETLOGON_VALIDATION_TICKET_LOGON message is used after the NETLOGON_TICKET_LOGON_INFO message (section 2.2.2.1) at the destination domain, the issuing KDC opens the ticket, verifies all the signatures, and then extracts the authorization information from the PAC. This message is defined with the following fields.



UserInformation (variable)
...
...
...
DeviceInformation (variable)
...
...
...
UserClaimsLength
UserClaims (variable)
...
...
...
DeviceClaimsLength
DeviceClaims

A - CriticalResults (1 byte): A UCHAR. Conditions that all parties must understand to interpret the rest of the results. The following value is defined.

Value	Meaning
LogonFailed 0x0000	There's no authorization data because the ticket logon failed. Check other result fields for the reason.

B - CriticalClientResults (1 byte): A UCHAR. Conditions that the caller must understand before using the results. Unused, MUST be set to 0.

C - CriticalComputerDomainResults (1 byte): A UCHAR. Conditions that must be handled by Netlogon in the computer's domain. Unused, MUST be set to 0.

D - CriticalTransitResults (1 byte): A UCHAR. Conditions that must be handled by Netlogon in every transited domain. Unused, MUST be set to 0.

SourceInformation (2 bytes): A USHORT that contains information about the ticket from the KDC that issued the service ticket. The following values are defined.

Value	Meaning
TicketDecryptionFailed 0x0000	Logon failed because the ticket could not be decrypted.
PacValidationFailed 0x0001	Logon failed because the PAC signatures did not validate.
CompoundSource 0x0002	The source ticket contained device information.
SourceUserClaims 0x0003	There were user claims in the source ticket.
SourceDeviceClaims 0x0004	There were device claims in the source ticket.
FullSignaturePresent 0x0005	The KDC checked the full ticket krbtgt signature.
ResourceGroupsRemoved 0x0006	The KDC removed (by client request) resource groups from the source information.

TransitInformation (2 bytes): A USHORT. Information from Netlogon about operations performed while transiting back to the computer. The following values are defined.

Value	Meaning
UserSidsFailed 0x0000	Logon failed because SID filtering did not allow the user identity.
UserNamespaceFailed 0x0001	Logon failed because namespace filtering did not allow the user domain name.
UserFailedA2A 0x0002	Logon failed because the user is not allowed to authenticate to the computer.
DeviceSidsFailed 0x0003	Compound identity was removed because SID filtering did not allow the device identity.
DeviceNamespaceFailed 0x0004	Compound identity was removed because SID filtering did not allow the device domain name.
UserSidsFiltered 0x0005	SID filtering removed one or more SIDs from the user information.
DeviceSidsFiltered 0x0006	SID filtering removed one or more SIDs from the device information.

KerberosStatus (4 bytes): A USHORT. If unsuccessful, includes an NTSTATUS code that details an error encountered by the KDC during ticket validation.

NetlogonStatus (4 bytes): A USHORT. If unsuccessful, includes an NTSTATUS code that details an error encountered by Netlogon during transit back to the computer.

UserInformation (variable): A NETLOGON_VALIDATION_SAM_INFO4 structure that contains the authenticated user information ([MS-NRPC] section 2.2.1.4.13).

DeviceInformation (variable): A NETLOGON_VALIDATION_SAM_INFO4 structure that contains optional authenticated device information ([MS-NRPC] section 2.2.1.4.13).

UserClaimsLength (4 bytes): A ULONG. The length of the preceding user claims data.

UserClaims (variable): A pointer to a UCHAR. The user claims data.

DeviceClaimsLength (4 bytes): A ULONG. The length of the preceding device claims data.

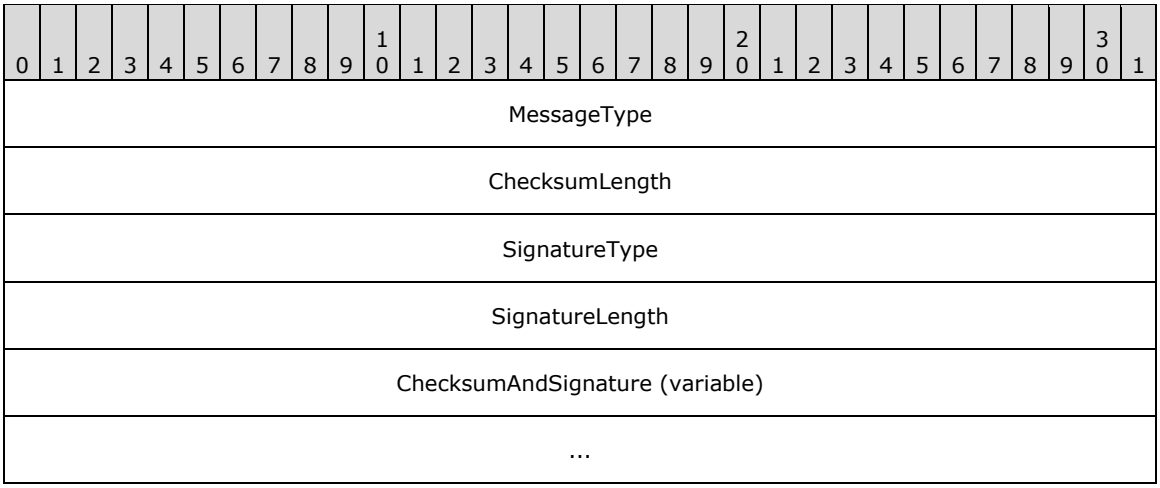
DeviceClaims (4 bytes): A pointer to a UCHAR. The device claims data.

2.2.4 Kerberos PAC Validation Message Syntax

The **privilege attribute certificate (PAC)** validation request message, KERB_VERIFY_PAC_REQUEST (section 2.2.4.1), MUST be encoded as a contiguous buffer. The encoded data SHOULD<2> be sent by using the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1). The encoding of the KERB_VERIFY_PAC_REQUEST is specified in section 2.2.4.1.

2.2.4.1 KERB_VERIFY_PAC_REQUEST Message

The KERB_VERIFY_PAC_REQUEST Message used for **PAC** validation is defined as follows.



MessageType (4 bytes): An unsigned 32-bit value describing the message type. This member MUST be set to 0x00000003.

ChecksumLength (4 bytes): An unsigned 32-bit value that MUST contain the signature length of the **PAC_SIGNATURE_DATA Signature** value ([MS-PAC] section 2.8) for the Server Signature ([MS-PAC] section 2.8.1) in the privilege attribute certificate (PAC).

SignatureType (4 bytes): An unsigned 32-bit value that MUST contain the **PAC_SIGNATURE_DATA SignatureType** value for the **Key Distribution Center (KDC)** Signature ([MS-PAC] section 2.8.1) in the PAC.

SignatureLength (4 bytes): An unsigned 32-bit value that MUST contain the signature length of the PAC_SIGNATURE_DATA Signature value in the KDC Signature in the PAC.

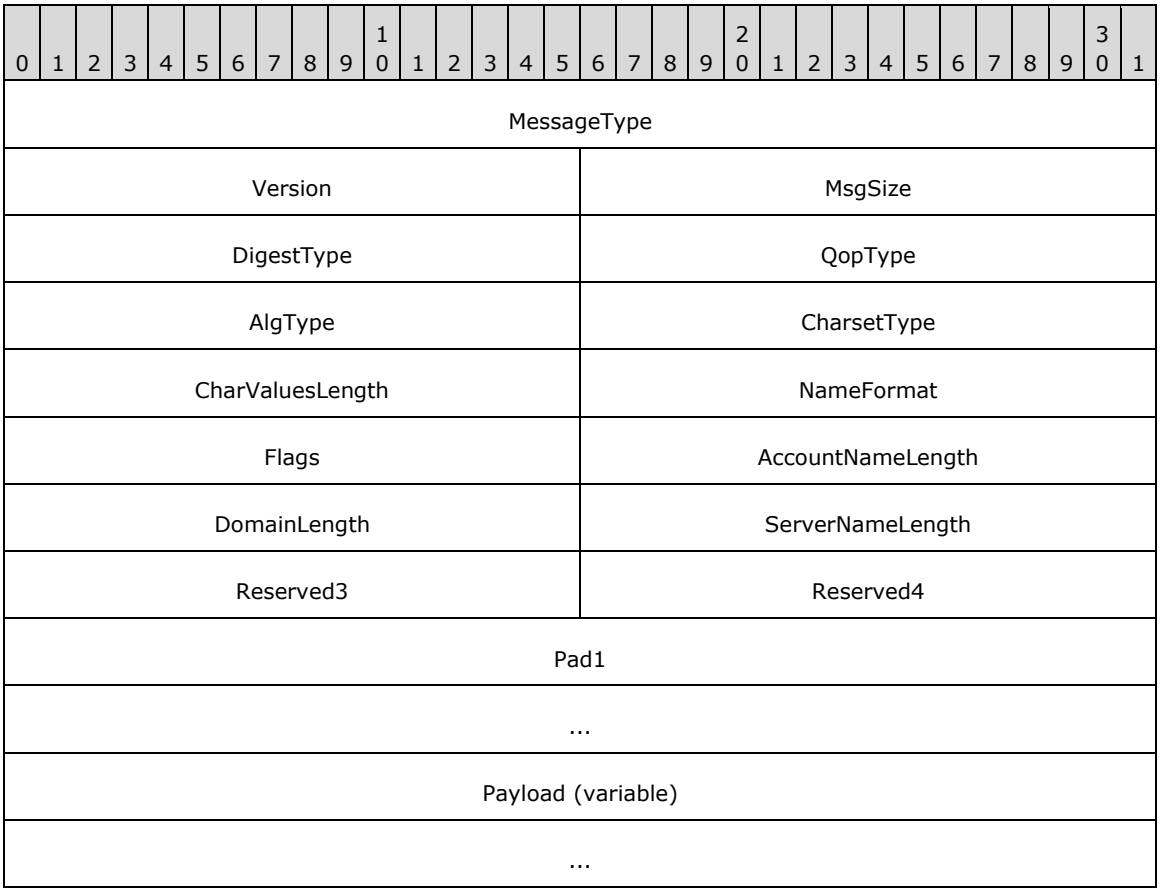
ChecksumAndSignature (variable): The **PAC_SIGNATURE_DATA Signature** value for the Server Signature in the PAC. It MUST be followed by the **PAC_SIGNATURE_DATA Signature** value for the KDC Signature in the PAC.

2.2.5 Digest Validation Message Syntax

The **Digest validation** protocol uses fields extracted from the digest-challenge and digest-response messages ([RFC2617] section 3.2 and [RFC2831] section 2.1) to verify the validity of the user signature (this is a hash performed with the user's password) and to retrieve the **PAC** for the user's account.

2.2.5.1 DIGEST_VALIDATION_REQ Message

The **DIGEST_VALIDATION_REQ** message defines a request to validate the input from the Digest Protocol Extensions [MS-DPSP] and retrieve user authorization information.



MessageType (4 bytes): A 32-bit unsigned integer that defines the **Digest validation** message type. This member MUST be set to 0x0000001A.

Version (2 bytes): A 16-bit unsigned integer that defines the version of the Digest validation protocol. The protocol version defined in this document is 1 (the value of this member MUST be 0x0001).

MsgSize (2 bytes): A 16-bit unsigned integer that MUST specify the total number of bytes in the **DIGEST_VALIDATION_REQ** message.

DigestType (2 bytes): A 16-bit unsigned integer that specifies the Digest protocol used, which MUST be one of the following:

Value	Meaning
0x0003	Using the **Digest authentication** mechanism [RFC2617] for the HTTP/1.1 Protocol.
0x0004	Using Digest authentication as a Simple Authentication and Security Layer (SASL) mechanism [RFC2831].

QopType (2 bytes): A 16-bit unsigned integer specifying the Quality of Protection (QoP) requested by the **Digest client** ([RFC2617] section 3.2.1 and [RFC2831] section 2.1.2.1) that MUST be one of the following:

Value	Meaning
0x0001	The Digest client did not specify a QoP. For backward compatibility with Digest Access authentication [RFC2069], Digest authentication made the QoP optional.
0x0002	Authentication only. Represents auth.
0x0003	Authentication and integrity protection. Represents auth-int.
0x0004	Authentication with integrity protection and encryption. Represents auth-conf.

AlgType (2 bytes): A 16-bit unsigned integer specifying the algorithm value specified by the Digest client in the digest-challenge message ([RFC2617] section 3.2.1 and [RFC2831]) that MUST be one of the following values:

Value	Meaning
0x0001	MD5 assumed; the algorithm was not present.
0x0002	MD5 value to produce the digest and checksum.
0x0003	MD5-sess value to produce the digest and checksum

CharsetType (2 bytes): A 16-bit unsigned integer specifying the type of encoding used for **username** and **password** fields that MUST be one of the following (as specified in [RFC2831] section 2.1.1 and [MS-DPSP] section 2.2):

Value	Meaning
0x0001	ISO8859-1 encoding is used for **username** and **password** fields.
0x0002	UTF-8 encoding is used for **username** and **password** fields.

CharValuesLength (2 bytes): A 16-bit unsigned integer that MUST specify the number of bytes in the **Payload** field of the **DIGEST_VALIDATION_REQ** message and MUST NOT exceed the total size in MsgSize.

NameFormat (2 bytes): A 16-bit unsigned integer specifying the format of the user **AccountName** field and MUST be one of the following (:

Value	Meaning
0x0000	**Digest server** cannot determine the format of the user's **AccountName**.
0x0001	A format determined to be the SAM account name ([MS-ADA3] 2.222).
0x0002	A format determined to be the **user principal name (UPN)** for the account ([MS-ADA3] 2.222).

Value	Meaning
0x0003	A format determined to be **NetBIOS** ([MS-ADA3] 2.4).

Flags (2 bytes): A two-byte set of bit flags providing additional instructions for processing the **DIGEST_VALIDATION_REQ** message by the **DC**. The **Flags** field is constructed from one or more bit flags from the following table, with the exception of the constraint on bit C.

Note All other bits MUST be set to zero and MUST be ignored upon receipt.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	E	D	C	B	A

A (1 bit): The format of **Username** and **Realm** (carried in the **Payload** field of **DIGEST_VALIDATION_REQ**) MUST be determined by the DC.

B (1 bit): The optional **Authzid** field ([RFC2831] section 2.1.2) is set and carried in the Payload buffer in the **DIGEST_VALIDATION_REQ** message.

C (1 bit): Indicates that this request is from a server, so group memberships are to be expanded for the Account's **PAC**. This bit MUST NOT be set if this request is forwarded from a server's **domain** to user account's domain.

D (1 bit): Indicates if a single backslash is found in the username value ([RFC2617] section 3.2.2).

E (1 bit): Indicates the DC will attempt to validate the request with an un-escaped backslash ([MS-DPSP] section 2.2).

AccountNameLength (2 bytes): A 16-bit unsigned integer that MUST specify the length of the **AccountName** field in the Payload buffer.

DomainLength (2 bytes): A 16-bit unsigned integer that MUST specify the length of the Domain field in the Payload buffer.

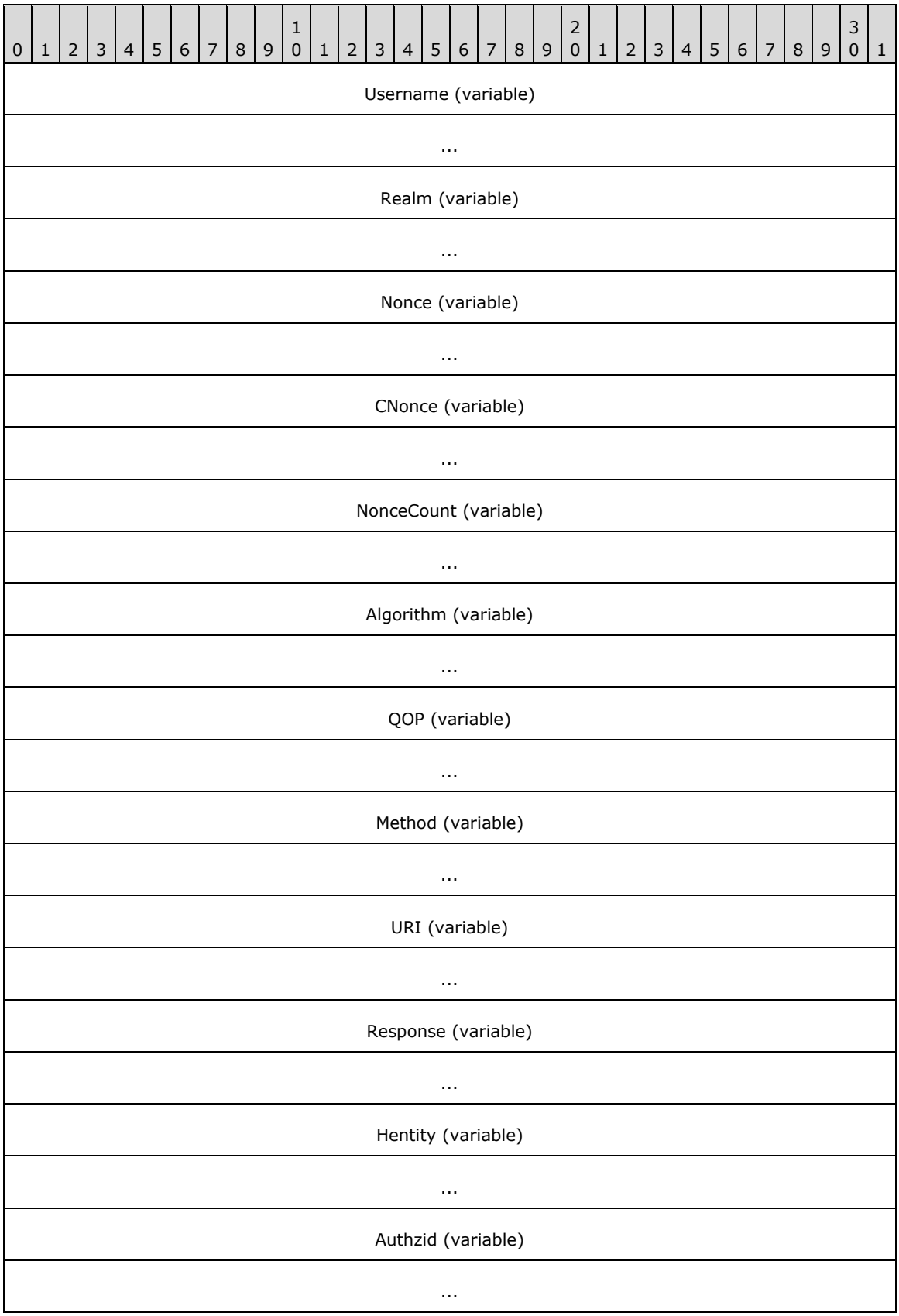
ServerNameLength (2 bytes): A 16-bit unsigned integer that MUST specify the length of the **ServerName** field in the Payload buffer.

Reserved3 (2 bytes): A 16-bit unsigned integer field reserved for future use. MUST be set to zero when sent and MUST be ignored on receipt.

Reserved4 (2 bytes): A 16-bit unsigned integer field reserved for future use. MUST be set to zero when sent and MUST be ignored on receipt.

Pad1 (8 bytes): An unused, 64-bit unsigned integer. MUST be set to zero when sent and MUST be ignored on receipt.

Payload (variable): A byte array that MUST contain the following strings in the following order. All strings are the unquoted directive value. All strings MUST be null-terminated; strings MUST be encoded by using [ISO/IEC-8859-1], unless specified as **Unicode**. Each of the strings MUST be included. If the string value is empty, then a terminating null character MUST be used for the value. Remember that the last three strings are Unicode strings, so they have a Unicode terminating null character.



AccountName (variable)
...
Domain (variable)
...
ServerName (variable)
...

Username (variable): The user name value from the digest-response message.
MUST be as specified in [RFC2617] section 3.2.2.

Realm (variable): The **realm** value.
MUST be as specified in [RFC2617] section 3.2.1.

Nonce (variable): The nonce value from the digest-challenge message.
MUST be as specified in [RFC2617] section 3.2.1.

CNonce (variable): The cnonce value from the digest-response message.
MUST be as specified in [RFC2617] section 3.2.2.

NonceCount (variable): The nc-value from the digest-response message.
MUST be as specified in [RFC2617], section 3.2.2.

Algorithm (variable): The algorithm value from the digest-response message.
MUST be as specified in [RFC2617] section 3.2.1.

QOP (variable): The QOP value from the digest-response message.
MUST be as specified in [RFC2617] section 3.2.2.

Method (variable): Method by which Digest authentication information MUST be transmitted as part of the HTTP1.1 protocol. The string value is GET or PUT if Digest authentication is used for the HTTP1.1 protocol. The string value is AUTHENTICATE if Digest authentication is used as an SASL mechanism [RFC2617].

URI (variable): The digest-URI value from the digest-response message.
MUST be as specified in [RFC2617] section 3.2.2.

Response (variable): The response value from the digest-response message.
MUST be as specified in [RFC2617] section 3.2.2.

Hentity (variable): The H (entity-body) value.
MUST be as specified in [RFC2617] section 3.2.2.3.

Authzid (variable): The Authzid value from the digest-response message.
MUST be as specified in [RFC2831] section 2.1.2.

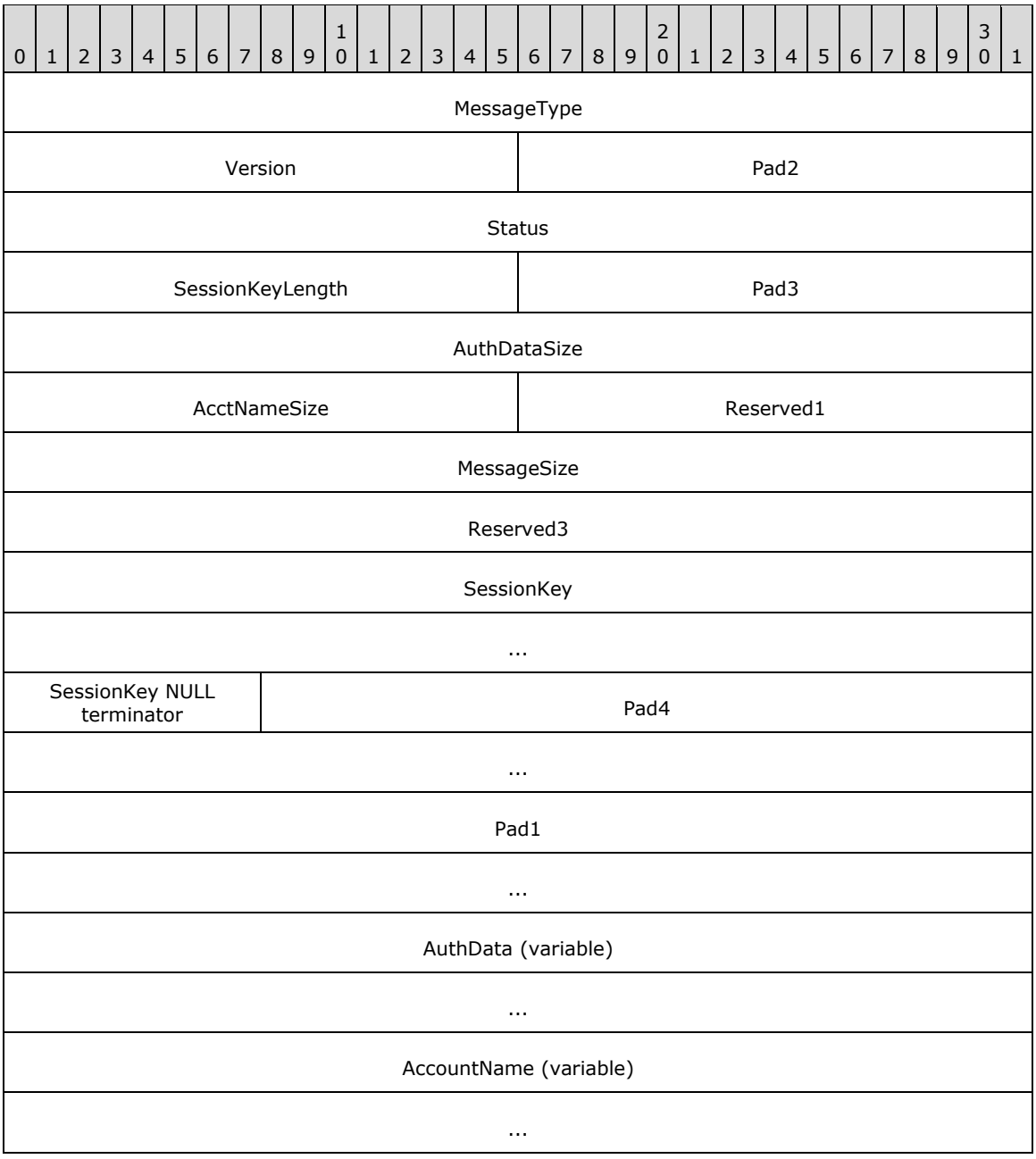
AccountName (variable): A Unicode string that MUST specify the user account name.

Domain (variable): A Unicode string that MUST specify the domain to which the user account belongs.

ServerName (variable): A Unicode string that MUST specify the NetBIOS name of the server that sent the DIGEST_VALIDATION_REQ message.

2.2.5.2 DIGEST_VALIDATION_RESP Message

The **DIGEST_VALIDATION_RESP** message is a response to a DIGEST_VALIDATION_REQ message (section 2.2.5.1).



MessageType (4 bytes): A 32-bit unsigned integer that MUST specify the **Digest validation** message type. This member MUST be 0x0000000A.

Version (2 bytes): A 16-bit unsigned integer that MUST specify the version of the Digest validation protocol. The protocol version defined in this document is 1. The value of this member MUST be 0x0001.

Pad2 (2 bytes): An unused 16-bit unsigned integer. MUST be set to zero when sent and MUST be ignored on receipt.

Status (4 bytes): A 32-bit unsigned integer that specifies if the **Digest authentication** data sent in the DIGEST_VALIDATION_REQ (section 2.2.5.1) was successfully verified by the **domain controller**. On successful validation, the **Status** field MUST be set to STATUS_SUCCESS. On failure, it MUST be set to STATUS_LOGON_FAILURE as specified in [MS-ERREF] section 2.3.

SessionKeyLength (2 bytes): A 16-bit unsigned integer that MUST specify the number of bytes of the **SessionKey** field in the DIGEST_VALIDATION_RESP message plus a terminating null character. It MUST be equal to 33.

Pad3 (2 bytes): An unused 16-bit unsigned integer. MUST be set to zero when sent and MUST be ignored on receipt.

AuthDataSize (4 bytes): A 32-bit unsigned integer that MUST specify the number of bytes of the **AuthData** field in the DIGEST_VALIDATION_RESP message.

AcctNameSize (2 bytes): A 16-bit unsigned integer that MUST specify the number of bytes of the **AccountName** field in the DIGEST_VALIDATION_RESP message.

Reserved1 (2 bytes): A 16-bit unsigned integer field reserved for future use. MUST be set to zero when sent and MUST be ignored on receipt.

MessageSize (4 bytes): A 32-bit unsigned integer that MUST specify the number of bytes in the entire DIGEST_VALIDATION_RESP message.

Reserved3 (4 bytes): A 32-bit unsigned integer field reserved for future use. MUST be set to zero when sent and MUST be ignored on receipt.

SessionKey (32 bytes): A 32-byte buffer that MUST contain the Digest SessionKey ([RFC2617] section 3.2.2.2).

SessionKey NULL terminator (1 byte): A single byte to terminate the SessionKey. MUST be set to zero.

Pad4 (7 bytes): An unused 7-byte padding. The value of each byte MUST be set to zero when sent and MUST be ignored on receipt.

Pad1 (8 bytes): An unused 64-bit unsigned integer. MUST be set to zero when sent and MUST be ignored on receipt.

AuthData (variable): This field MUST contain a **PACTYPE** structure ([MS-PAC] section 2.3). The length of the **PACTYPE** structure MUST be specified by the **AuthDataSize** field. The length of this field MUST be 0 if the value of the **Status** field is STATUS_LOGON_FAILURE.

AccountName (variable): This field MUST contain the **NetBIOS** name of the user's account. Its length MUST be specified in the **AcctNameSize** field.

3 Protocol Details

Authentication Protocol Domain Support (specified for each of the protocols in the following sections) uses NRPC [MS-NRPC] for transport. Each of the following protocols is a simple request-response exchange over this transport. The security assurances provided by the underlying Netlogon and **RPC** protocols are common to all of these protocols. All of these exchanges require that a remote procedure call (RPC) connection through the Netlogon secure channel MUST be established with a **domain controller (DC)** for the **domain** to which the server belongs (section 1.5).

3.1 NTLM Logon Details

NT LAN Manager (NTLM) **interactive logon** and **network logon** MUST complete the authentication sequence by contacting the **DC** using an NRPC pass-through authentication ([MS-NRPC] section 3.2) method, with parameters as specified in section 3.1.5.

The NETLOGON_NETWORK_INFO and NETLOGON_VALIDATION_SAM_INFO4 data structures SHOULD be exchanged ([MS-NRPC] sections 2.2.1.4.5 and 2.2.1.4.13). The DC MUST populate the NETLOGON_VALIDATION_SAM_INFO4 with the information for the user logging on and return the SessionBaseKey ([MS-NLMP] section 3.3.1 and 3.3.2) in the **UserSessionKey** field in NETLOGON_VALIDATION_SAM_INFO4, and MUST send it back to the **NTLM server**. If no matching account is found, an error STATUS_NO_SUCH_USER (section 2.2) MUST be returned to the NTLM server, resulting in a logon failure.

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The protocol requires that the **DC** MUST have a database or **directory** of accounts with authorization information available to it.

The **NTLM** abstract data model is specified in [MS-NLMP] section 3.1.1. The Netlogon abstract data model is specified in [MS-NRPC] section 3.1.1.

The **NTLM server** uses the following configuration values:

LogonAttempts: A 32-bit unsigned integer that contains the total number of logon attempts since the last restart.

NTLMServerDomainBlocked: A Boolean setting that SHOULD<3> control the NTLM server that is responding to NTLM authentication requests. When set to TRUE, this setting disables the NTLM server from sending NTLM pass-through authentication messages (section 3.1.5) to any DC.

For NTLM server implementations that use an authorization model that is based on a security identifier (SID), the server maintains the following parameter for each security context:

ImpersonationAccessToken (Public): A Token/Authorization Context (see [MS-DTYP] section 2.5.2).

The DC SHOULD<4> use the following configuration values:

AccountDCBlocked: A Boolean setting that controls the DC responding to NTLM authentication requests. When set to TRUE, this setting disables the account **domain** DC from responding to NTLM pass-through authentication messages (section 3.1.5).

ResourceDCBlocked: A Boolean setting that controls the DC responding to NTLM authentication requests. When set to TRUE, this setting disables the resource domain DC from sending NTLM pass-through authentication messages (section 3.1.5).

DCBlockExceptions: A list of server names that can use NTLM authentication.

The NTLM server MAY<5> use the following configuration value:

AllowComputerLogon: A Boolean setting that indicates that the caller wants to authenticate a computer. Setting this flag results in the **K** bit being set in **LogonInformation.LogonNetwork.Identity.ParameterControl.**

3.1.2 Timers

None.

3.1.3 Initialization

The **LogonAttempts** field SHOULD be initialized to zero on startup.

3.1.4 Higher-Layer Triggered Events

The **NTLM** logon message exchange MUST be triggered by a server requesting user authentication via the Netlogon **remote procedure call (RPC)** mechanism to the **domain controller (DC)**.

3.1.5 Message Processing Events and Sequencing Rules

NTLM logon is a stateless protocol with request-response semantics.

The **NTLM server** MAY<6> call the **NetrLogonSamLogonEx** method ([MS-NRPC] section 3.5.4.5.1) with the parameters defined in the following sections. Based on the account name supplied, a **domain controller (DC)** for the **domain** MUST be located ([MS-ADTS] section 6.3.6). The NTLM server MUST establish a connection with the DC ([MS-NRPC] section 3.1.4.6). The NTLM server SHOULD invoke the NetrLogonSamLogonEx method ([MS-NRPC] section 3.5.4.5.1).

If **NTLMServerDomainBlocked** == TRUE, the NTLM server SHOULD<7> return STATUS_NTLM_BLOCKED to the **NTLM client**.

If the DC is of the resource domain:

- If **ResourceDCBlocked** == TRUE, and the NTLM server's name is not equal to any of the **DCBlockExceptions** server names, the DC SHOULD<8> return STATUS_NTLM_BLOCKED.

If the DC is of the account domain:

- If **AccountDCBlocked** == TRUE, the **APDS server** SHOULD<9> return STATUS_NTLM_BLOCKED.
- If the **domainControllerFunctionality** attribute ([MS-ADTS] section 3.1.1.3.2.25) returns a value that is >= 6, the account is not also the NTLM server's account, and the APDS server determines that an authentication policy setting ([MS-KILE] section 3.3.5.5) applies, then:
  - If **AllowedToAuthenticateTo** is not NULL, an access check SHOULD<10> be performed to determine whether the user has the ACL granting ACTRL_DS_CONTROL_ACCESS ([MS-SAMR] section 2.2.1.17). If the access check fails, APDS MUST return STATUS_AUTHENTICATION_FIREWALL_FAILED.

The DC MUST verify the account access status. If the account is not valid for logon, the APDS server returns one of the following errors:

- If the **userAccountControl** attribute ([MS-ADTS] section 2.2.16) **D** flag is set to TRUE, the APDS server returns STATUS_ACCOUNT_DISABLED.
- If the **AccountExpires** attribute ([MS-ADA1] section 2.1) is set to a value that is in the past, the APDS server returns STATUS_ACCOUNT_EXPIRED.
- If the **userAccountControl** attribute ([MS-ADTS] section 2.2.16) **L** flag is set to TRUE, the APDS server returns STATUS_ACCOUNT_LOCKED_OUT.
- If the current time is not within **logonHours** attribute ([MS-ADA1] section 2.376), the APDS server returns STATUS_INVALID_LOGON_HOURS.
- If **PasswordMustChange**, which is generated with the same method as specified in [MS-SAMR] section 3.1.5.14.4, is set to a value that is in the past, the APDS server returns STATUS_PASSWORD_EXPIRED.
- If **PasswordMustChange**, ([MS-SAMR] section 3.1.5.14.4), is zero, the APDS server returns STATUS_PASSWORD_MUST_CHANGE.
- If the **userAccountControl** attribute ([MS-ADTS] section 2.2.16) **SR** flag is set to TRUE, because this is a password-based logon, the APDS server returns STATUS_SMARTCARD_LOGON_REQUIRED.
- If the **userAccountControl** attribute ([MS-ADTS] section 2.2.16) **ID** flag is set to TRUE, the APDS server returns STATUS_NOLOGON_INTERDOMAIN_TRUST_ACCOUNT.
- If the **userAccountControl** attribute ([MS-ADTS] section 2.2.16) **WT** flag is set to TRUE, the APDS server returns STATUS_NOLOGON_WORKSTATION_TRUST_ACCOUNT.
- If the **userAccountControl** attribute ([MS-ADTS] section 2.2.16) **ST** flag is set to TRUE, the APDS server returns STATUS_NOLOGON_SERVER_TRUST_ACCOUNT.

An APDS server implementation can choose to send more descriptive error codes (as in the case above). However, the NTLM server MUST treat any error returned by the DC as a logon failure.

The DC attempts to validate the request, increment **LogonAttempts**, and if successful, authenticate the user. If validation is unsuccessful, the DC MUST return an error. The role of the DC in the NTLM authentication sequence is specified in [MS-NLMP] section 3.3.

Upon successful validation:

- If the **domainControllerFunctionality** attribute ([MS-ADTS] section 3.1.1.3.2.25) returns a value that is >= 6 and the user is a member of PROTECTED_USERS ([MS-DTYP] section 2.4.2.4), APDS SHOULD<11> return STATUS_ACCOUNT_RESTRICTION.
- Otherwise, the user account's DC MUST send the domain global groups and universal groups (that the user is a member of) to the server's DC, and MUST follow the trust path that was used to contact the user's account DC ([MS-NRPC] section 3.5.4.5.1).

When the trust crossed in the trust path has the TRUST_ATTRIBUTE_CROSS_ORGANIZATION ([MS-LSAD] section 2.2.7.9) set, the DC MUST add the OTHER_ORGANIZATION SID ([MS-DTYP] section 2.4.2.4) to the user's groups.

When a user has the OTHER_ORGANIZATION SID, the server domain DC MUST perform an access check where:

- The security descriptor MUST contain the ACL granting the client user ACTRL_DS_CONTROL_ACCESS ([MS-SAMR] section 2.2.1.17) to the **server computer's** AD account object.

If the access check fails, the DC MUST reject the authentication request and return STATUS_AUTHENTICATION_FIREWALL_FAILED. The server domain DC also MUST add the domain local groups, and then send the entire list of groups to the NTLM server to be used for authorization decisions.

For NTLM server implementations that use an authorization model that is based on a security identifier (SID), the server SHOULD populate the User SID and Security Group SIDs in the **ImpersonationAccessToken** (section 3.1.1) as follows:

- Concatenate **LogonDomainId** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13) and **UserId** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13), add the result to the **ImpersonationAccessToken.Sids** array, and set the **ImpersonationAccessToken.UserIndex** field to this index.
- Concatenate **LogonDomainId** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13) and **PrimaryGroupId** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13), add the result to the **ImpersonationAccessToken.Sids** array, and set the **ImpersonationAccessToken.PrimaryGroup** field to this index.
- For each **GroupIds** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13), concatenate **LogonDomainId** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13) and **GroupIds.RelativeID** ([MS-NRPC] sections 2.2.1.4.11, 2.2.1.4.12, and 2.2.1.4.13), and add the result to the **ImpersonationAccessToken.Sids** array.
- For each **ExtraSids** ([MS-NRPC] sections 2.2.1.4.12 and 2.2.1.4.13), add the **ExtraSids.Sid** ([MS-NRPC] sections 2.2.1.4.12 and 2.2.1.4.13) to the **ImpersonationAccessToken.Sids** array.

The server calls GatherGroupMembershipForSystem ([MS-DTYP] section 2.5.2.1.1), where **InitialMembership** contains the **ImpersonationAccessToken.Sids** array, and set the **ImpersonationAccessToken.Sids** array to **FinalMembership**.

The server calls AddPrivilegesToToken ([MS-DTYP] section 2.5.2.1.2), where **Token** contains **ImpersonationAccessToken**.

Other SID structures can be added to **ImpersonationAccessToken** following authentication (see [MS-DTYP] section 2.7.1).

3.1.5.1 NTLM Interactive Logon

If the **domainControllerFunctionality** attribute ([MS-ADTS] section 3.1.1.3.2.25) returns a value that is >= 6, the account is not also the NTLM server's account, and the APDS server determines that an authentication policy setting ([MS-KILE] section 3.3.5.5) applies:

- If the account is:
  - A user account object, and the corresponding **msDS-UserAllowedToAuthenticateFrom** attribute ([MS-ADA2] section 2.492) is populated, APDS SHOULD<12> return STATUS_ACCOUNT_RESTRICTION.
  - A managed Service account object, and the corresponding **msDS-ServiceAllowedToAuthenticateFrom** ([MS-ADA2] section 2.460) is populated, APDS SHOULD<13> return STATUS_ACCOUNT_RESTRICTION.

For **NTLM interactive logons**, the **NTLM server** MAY<14> call NetrLogonSamLogonEx ([MS-NRPC] section 3.5.4.5.1) with the following parameters (set as specified):

- **LogonLevel** MUST be **NetlogonInteractiveInformation**.
- IF the **G** flag in *NegotiateFlags* ([MS-NRPC] section 3.1.4.2) is set to FALSE, the *ValidationLevel* MUST be NetlogonValidationSamInfo ([MS-NRPC] section 2.2.1.4.17).

  ELSE IF the **Y** or **T** flags are set to FALSE in *NegotiateFlags* ([MS-NRPC] section 3.1.4.2), the *ValidationLevel* MUST be NetlogonValidationSamInfo2 ([MS-NRPC] section 2.2.1.4.17).

  ENDIF.
- IF SealSecureChannel ([MS-NRPC] section 3.1.1) is set to FALSE, the *ValidationLevel* MUST be NetlogonValidationSamInfo2 ([MS-NRPC] section 2.2.1.4.17).

  ELSE the *ValidationLevel* SHOULD<15> be NetlogonValidationSamInfo4 ([MS-NRPC] section 2.2.1.4.17).

  ENDIF.
- *LogonInformation* MUST contain a reference to NETLOGON_INTERACTIVE_INFO ([MS-NRPC] section 2.2.1.4.3).

The logon request MUST be sent to the **domain controller** of the user account **domain** that has been located.

If the domain controller for the user account is not reachable, but the user domain is one of the trusted domains, the logon MUST fail. If the user domain is not one of the trusted domains, the NTLM server's local account database MUST be used to authenticate the user.

The request that is sent to the user account domain controller MUST contain the **NTOWF** of the user's password.

The domain controller MUST verify the response to the challenge ([MS-NLMP] section 3.3). If there is a successful match, the domain controller MUST return data with *ValidationInformation* containing a reference to:

- NETLOGON_VALIDATION_SAM_INFO4 ([MS-NRPC] section 2.2.1.4.13), if the *ValidationLevel* in the request is **NetlogonValidationSamInfo4**.
- NETLOGON_VALIDATION_SAM_INFO2 ([MS-NRPC] section 2.2.1.4.12), if the ValidationLevel in the request is **NetlogonValidationSamInfo2**.
- NETLOGON_VALIDATION_SAM_INFO ([MS-NRPC] section 2.2.1.4.11), if the ValidationLevel in the request is **NetlogonValidationSamInfo**.

If there is not a match, the DC SHOULD<16> return the failure error code STATUS_WRONG_PASSWORD (section 2.2) with no response data.

3.1.5.2 NTLM Network Logon

If the domainControllerFunctionality attribute ([MS-ADTS] section 3.1.1.3.2.25) returns a value that is >= 6, the account is not also the NTLM server's account, and the APDS server determines that an authentication policy setting ([MS-KILE] section 3.3.5.5) applies:

1. If the **domainControllerFunctionality** attribute ([MS-ADTS] section 3.1.1.3.2.25) returns a value that is < 7, the msDS-UserAllowedNTLMNetworkAuthentication and **msDS-ServiceAllowedNTLMNetworkAuthentication** attributes ([MS-ADA2] section 2.491 and [MS-ADA2] section 2.459, respectively) SHOULD<17> be treated as set to FALSE.

2. If a user account object, and if the corresponding **msDS-UserAllowedToAuthenticateFrom** ([MS-ADA2] section 2.492) is populated and **msDS-UserAllowedNTLMNetworkAuthentication** is set to FALSE, APDS MUST return STATUS_ACCOUNT_RESTRICTION.

3. If a managed Service account object, and if the corresponding **msDS-ServiceAllowedToAuthenticateFrom** ([MS-ADA2] section 2.460) is populated and **msDS-ServiceAllowedNTLMNetworkAuthentication** is set to FALSE, APDS MUST return STATUS_ACCOUNT_RESTRICTION.

For **NTLM network logons**, the **NTLM server** MAY<18> call **NetrLogonSamLogonEx** ([MS-NRPC] section 3.5.4.5.1) with the following parameters (set as specified):

- **LogonLevel** MUST be **NetlogonNetworkInformation**.
- IF the **G** flag in *NegotiateFlags* ([MS-NRPC] section 3.1.4.2) is set to FALSE, the *ValidationLevel* MUST be NetlogonValidationSamInfo ([MS-NRPC] section 2.2.1.4.17).

  ELSE IF the **Y** or **T** flags in *NegotiateFlags* ([MS-NRPC] section 3.1.4.2) are set to FALSE, the *ValidationLevel* MUST be NetlogonValidationSamInfo2 ([MS-NRPC] section 2.2.1.4.17).

  ENDIF.
- IF SealSecureChannel ([MS-NRPC] section 3.1.1) is set to FALSE, the *ValidationLevel* MUST be NetlogonValidationSamInfo2 ([MS-NRPC] section 2.2.1.4.17).

  ELSE the *ValidationLevel* SHOULD<19> be NetlogonValidationSamInfo4 ([MS-NRPC] section 2.2.1.4.17).

  ENDIF.
- *LogonInformation* MUST contain a reference to NETLOGON_NETWORK_INFO ([MS-NRPC] section 2.2.1.4.5).
- Set the **E** and **K** bits of **LogonInformation.LogonNetwork.Identity.ParameterControl**.<20>
- The following algorithm is used for authentication from the server to the **DC**:

  IF (NTLMSSP_NEGOTIATE_ENHANCED_SESSION_SECURITY and NtResponseLength == 24 and LmResponseLength >= 8)

  - NetlogonNetworkInformation.LmChallenge = MD5(Concatenate(ChallengeToClient, LmResponse[0..7]))[0..7]

  ELSE

  - NetlogonNetworkInformation.LmChallenge = ChallengeToClient

  END

The DC of the server's **domain** MUST be located ([MS-NRPC] section 3.5.4.3) and the request sent to it. This request MUST contain the NTLM challenge-response pair that was exchanged between the NTLM server and the client ([MS-NLMP] sections 2.2.1.2 and 2.2.1.3).

The DC verifies the response to the challenge either as defined in [MS-NLMP] section 3.3 or by using a subauthentication package (section 3.1.5.2.1).

If the account is a computer account, the subauthentication package is not verified, and the **K** bit of **LogonInformation.LogonNetwork.Identity.ParameterControl** is not set, then return STATUS_NOLOGON_WORKSTATION_TRUST_ACCOUNT.<21>

If the account is a domain controller computer account, the subauthentication package is not verified, and the E bit of **LogonInformation.LogonNetwork.Identity.ParameterControl** is not set, return STATUS_NOLOGON_SERVER_TRUST_ACCOUNT.

For NTLMv2 authentication [MS-NLMP], the DC MUST verify that the request originated from the NTLM server that generated the challenge:

1. The DC extracts the MsvAvNbComputerName and MsvAvNbDomainName AV pairs ([MS-NLMP] section 2.2.2.1) from the NTLMv2_CLIENT_CHALLENGE ([MS-NLMP] section 2.2.2.7) of the AUTHENTICATE_MESSAGE ([MS-NLMP] section 2.2.1.3).

2. If MsvAvNbDomainName does not match the **NetBIOS** name of the DC's domain, then return STATUS_LOGON_FAILURE (section 2.2).

3. If MsvAvNbComputerName does not match the NetBIOS name of the server that established the secure channel ([MS-NRPC] section 3.5.4.4.2), then return STATUS_LOGON_FAILURE.

If there is a match, the DC MUST return data with ValidationInformation containing a reference to NETLOGON_VALIDATION_SAM_INFO4 ([MS-NRPC] section 2.2.1.4.13, if the ValidationLevel in the request is NetlogonValidationSamInfo4) or a reference to NETLOGON_VALIDATION_SAM_INFO2 ([MS-NRPC] section 2.2.1.4.12, if the ValidationLevel in the request is NetlogonValidationSamInfo2) or a reference to NETLOGON_VALIDATION_SAM_INFO ([MS-NRPC] section 2.2.1.4.11, if the ValidationLevel in the request is NetlogonValidationSamInfo). If there is not a match, the DC MUST return a failure error code STATUS_LOGON_FAILURE with no response data.<22>

3.1.5.2.1 Verifying Responses with Sub-Authentication Packages

The request to verify by a subauthentication package SHOULD<23> be indicated by the **ParameterControl** field of the *LogonInformation* parameter. The **ParameterControl** field, defined in [MS-NRPC] section 2.2.1.4.15, provides an extensibility point for software providers.

An example of subauthentication package usage occurs with remote access authentications using the CHAP ([RFC1994]) method. In such cases, response computation of a MD5 hash value ([RFC1994]) is used instead of **NTOWF**.

Using the NRPC generic pass-through ([MS-NRPC] section 3.2.4.1) can result in invoking a custom subauthentication package at the **DC**, per the indication of the **ParameterControl** field of the *LogonInformation* parameter. In this case, such a subauthentication package MUST serve as a custom response verification instead of using the method specified by section 3.3 of [MS-NLMP]. For more information about this subauthentication package, see [MSDN-0SUBAUTHROUTINE9].

3.1.6 Timer Events

There are no timer events for **NTLM** logon. All associated timer events are specified in the Netlogon Remote Protocol ([MS-NRPC]) that serves as the transport.

3.1.7 Other Local Events

None.

3.2 Kerberos PAC Validation Details

Kerberos **PAC** validation SHOULD use the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1). The NETLOGON_TICKET_LOGON_INFO message (section 2.2.2.1) MUST be sent to the **domain controller (DC)** for privilege attribute certificate (PAC) verification. The ticket verification algorithm MUST occur (section 3.2.5).

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The protocol requires that the **DC** MUST have a database or **directory** of accounts with **Kerberos** keys ([MS-KILE] section 3.1.1.2) available to it.

The server operating system MUST have the **PAC**.

The Netlogon abstract data model is specified in [MS-NRPC] section 3.2.1.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

For information on higher-layer triggered events, see section 1.6.2.

3.2.5 Message Processing Events and Sequencing Rules

Kerberos PAC validation is a stateless protocol with request-response semantics.

3.2.5.1 Generating a NETLOGON_TICKET_LOGON_INFO Message

The server operating system MUST first assemble the NETLOGON_TICKET_LOGON_INFO message structure (section 2.2.2.1) by copying the service Ticket that the server operating system is to verify. The message type field MUST be set to 0x00000026 to make the server operating system ready to contact the **DC**.

This exchange MUST be layered on top of the Netlogon generic pass-through ([MS-NRPC] section 3.2.4.1). The server operating system MUST supply a **NETLOGON_TICKET_LOGON_INFO** structure, packed as a single buffer, as the **LogonData** field. The **PackageName** field MUST be set to a UNICODE_STRING with a buffer of **Kerberos**.

If the DC cannot be reached, Netlogon ([MS-NRPC] section 3) MUST return the error STATUS_NO_LOGON_SERVERS (section 2.2). The server operating system SHOULD fail the authentication attempt.

3.2.5.2 Processing a NETLOGON_TICKET_LOGON_INFO Message

On receipt of the message, the **DC** MUST decode the NETLOGON_TICKET_LOGON_INFO message (section 2.2.2.1). The DC MUST decrypt the ticket and decode the PAC to extract the Ticket checksum, KDC checksum, and server checksum. The DC MUST verify the KDC checksum, which is a keyed hash [RFC4757] over the server checksum passed in the request. If the checksum verification fails, the DC MUST return an error code, STATUS_LOGON_FAILURE (section 2.2) as the return value to the Netlogon Generic Pass-through method. If the checksum is verified, the DC MUST return STATUS_SUCCESS. There is no return message.

3.2.5.3 Generating a KERB_VERIFY_PAC_REQUEST Message

The server operating system MUST first assemble the KERB_VERIFY_PAC_REQUEST (section 2.2.4.1) message structure by copying the signature values out of the **privilege attribute certificate (PAC)** ([MS-PAC] section 2.8) that the server operating system is verifying. The message type field MUST be set to 0x00000003 to make the server operating system ready to contact the **DC**.

This exchange MUST be layered on top of the Netlogon generic pass-through ([MS-NRPC] section 3.2.4.1). The server operating system MUST supply a **KERB_VERIFY_PAC_REQUEST** structure, packed as a single buffer, as the **LogonData** field. The **PackageName** field MUST be set to a UNICODE_STRING with a buffer of **Kerberos**.

If the DC cannot be reached, Netlogon ([MS-NRPC] section 3) MUST return the error STATUS_NO_LOGON_SERVERS (section 2.2). The server operating system SHOULD fail the authentication attempt.

3.2.5.4 Processing a KERB_VERIFY_PAC_REQUEST Message

On receipt of the message, the **DC** MUST decode the KERB_VERIFY_PAC_REQUEST (section 2.2.4.1) message to locate the server checksum and the **Key Distribution Center (KDC)** checksum values. The DC MUST verify the KDC checksum, which is a keyed hash [RFC4757] over the server checksum passed in the request. If the checksum verification fails, the DC MUST return an error code, STATUS_LOGON_FAILURE (section 2.2) as the return value to the Netlogon Generic Pass-through method. If the checksum is verified, the DC MUST return STATUS_SUCCESS. There is no return message.

When the method completes, the server operating system MUST examine the return code to determine if the **PAC** contents has been altered. Any nonzero return code MUST be treated by the server operating system as a failure.

3.2.6 Timer Events

There are no timer events for **Kerberos PAC** validation. All associated timer events are specified in the Netlogon Remote Protocol [MS-NRPC] that serves as the transport for Kerberos PAC validation messages.

3.2.7 Other Local Events

There are no other local events that impact the operation of this protocol.

3.3 Digest Validation Details

The **Digest validation** protocol SHOULD<24> use the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1). The exchanged messages are DIGEST_VALIDATION_REQ (section 2.2.5.1) and DIGEST_VALIDATION_RESP (section 2.2.5.2). The DIGEST_VALIDATION_RESP message MUST contain the status of authentication validation and, on success, MUST also contain the user's authorization information.

3.3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The protocol requires that the **DC** MUST have a database or directory of accounts with authentication and authorization information available to it. All state information, specifically the nonce challenge that foils replay attacks ([MS-DPSP] section 3.2.1) MUST be handled by the participants of the **Digest authentication** protocol [RFC2617] [RFC2831] and MUST be sent as part of the DIGEST_VALIDATION_REQ message (section 2.2.5.1).

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

The **Digest validation** message exchange is triggered by a server that is configured to authenticate users by using the **Digest authentication**. During the Digest authentication exchange (as specified in [RFC2617] and [RFC2831]), the **Digest server** initiates the Digest validation exchange with the **DC**. This transaction does not have direct access to the user's password and relies on the DC to validate the digest input data sent by the client.

3.3.5 Message Processing Events and Sequencing Rules

Digest validation is a stateless protocol with request-response semantics. The general model is as follows:

1. After the **Digest server** sends the DIGEST_VALIDATION_REQ (section 2.2.5.1) message, it MUST receive either a DIGEST_VALIDATION_RESP (section 2.2.5.2) message from the **DC** or an error status in the Netlogon generic pass-through function ([MS-NRPC] section 3.2.4.1).

2. Upon receiving the DIGEST_VALIDATION_REQ message, the DC MUST verify the keyed hash contained in the Payload buffer's **Response** field, and then send the DIGEST_VALIDATION_RESP message to the Digest server.

3.3.5.1 Generating the DIGEST_VALIDATION_REQ Message

The **Digest server** MUST construct the DIGEST_VALIDATION_REQ (section 2.2.5.1) message by using fields extracted from the digest-challenge and digest-response messages ([RFC2617] section 3.2 and [RFC2831] section 2.1). This message MUST be sent to the **DC** to verify the validity of the user's signature (keyed hash performed with the user's password) and to retrieve the **privilege attribute certificate (PAC)** for the user's account. If the DC cannot be contacted for any reason, the Digest server fails the authentication attempt.

The DIGEST_VALIDATION_REQ message MUST be packed as a contiguous buffer, and the encoded data sent by using the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1). The encoding of the DIGEST_VALIDATION_REQ is as specified in section 2.2.5.1. The PackageName ([MS-NRPC] section 3.2.4.1) in the NETLOGON_GENERIC_INFO structure ([MS-NRPC] section 2.2.1.4.2) MUST be WDigest. The **AlgType** field of the DIGEST_VALIDATION_REQ (section 2.2.5.1) message SHOULD<25> be set to 0x03.

3.3.5.2 Request Processing and Generating DIGEST_VALIDATION_RESP Message

If the **AlgType** field of the DIGEST_VALIDATION_REQ (section 2.2.5.1) is not set to 0x03, the **DC** SHOULD<26> return SEC_E_QOP_NOT_SUPPORTED.

Using the **Username** field in the **Payload** field of the DIGEST_VALIDATION_REQ (section 2.2.5.1) message, the DC MUST look up the user's password. If the account is not found and Bit A of the **Flags** field of the DIGEST_VALIDATION_REQ message is set to:

0: The DC SHOULD return STATUS_NO_SUCH_USER.

1: If the domain name in the **Realm** field matches the DC's domain name, then fail with STATUS_LOGON_FAILURE. Otherwise, using the **Realm** field in the **Payload** field of the DIGEST_VALIDATION_REQ message to determine the **domain**, the DC SHOULD send the DIGEST_VALIDATION_REQ message to a DC in that domain.

The DC MUST verify the keyed hash contained in the Payload buffer's **Response** field of the DIGEST_VALIDATION_REQ (section 2.2.5.1) message. The algorithm to perform this validation MUST be as specified in [RFC2617] section 3.2.2.1 and [RFC2831] section 2.1.2.1.

If validation is successful, a DIGEST_VALIDATION_RESP (section 2.2.5.2) message with Status [MS-ERREF] indicating successful authentication (that is, STATUS_SUCCESS) and authorization information for the user's account (the **PAC**) MUST be sent back to the **Digest server**. If unsuccessful, the DC MUST return an error code as an error status in NRPC API. It MUST NOT send back the DIGEST_VALIDATION_RESP message.

The **Digest validation** response message DIGEST_VALIDATION_RESP MUST be packed as a contiguous buffer, and the encoded data SHOULD be sent by using the generic pass-through mechanism ([MS-NRPC] section 3.2.4.1). The encoding of DIGEST_VALIDATION_RESP is as specified in section 2.2.5.2. The Digest validation response message is sent by using the generic pass-through mechanism, as specified in [MS-NRPC] section 3.2.4.1.

3.3.6 Timer Events

There are no timer events for the **Digest validation** protocol. All associated timer events are specified in the Netlogon Remote Protocol ([MS-NRPC]) that serves as the transport for Digest validation messages.

3.3.7 Other Local Events

There are no other local events that impact the operation of this protocol.

4 Protocol Examples

4.1 NTLM Pass-Through Authentication

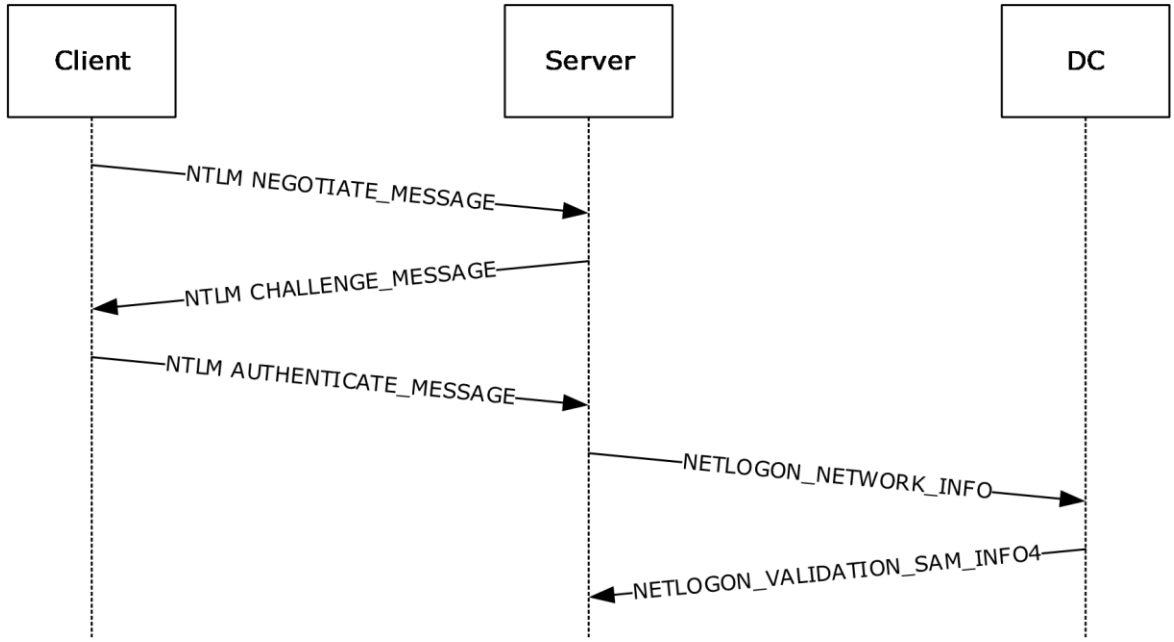


Figure 2: NTLM pass-through authentication

1. The user logs on to the computer desktop (labeled Client) by typing in the user name and password. Client sends an **NTLM** NEGOTIATE_MESSAGE ([MS-NLMP] section 2.2.1.1) to request authentication to the server.

2. The server sends an NTLM CHALLENGE_MESSAGE ([MS-NLMP] section 2.2.1.2) to the client.

3. The client responds to the challenge by signing it with its key and sending the response in an NTLM AUTHENTICATE_MESSAGE ([MS-NLMP] section 2.2.1.3) to the server.

4. The server forwards the client's response to the **domain controller** in a NETLOGON_NETWORK_INFO message.

5. The domain controller verifies the signature on the response, and returns the result to the server in a NETLOGON_VALIDATION_SAM_INFO4 message. If the verification is successful, the message contains the user's **PAC** with the authorization data. If the verification is unsuccessful, logon is denied.

4.2 Kerberos PAC Validation

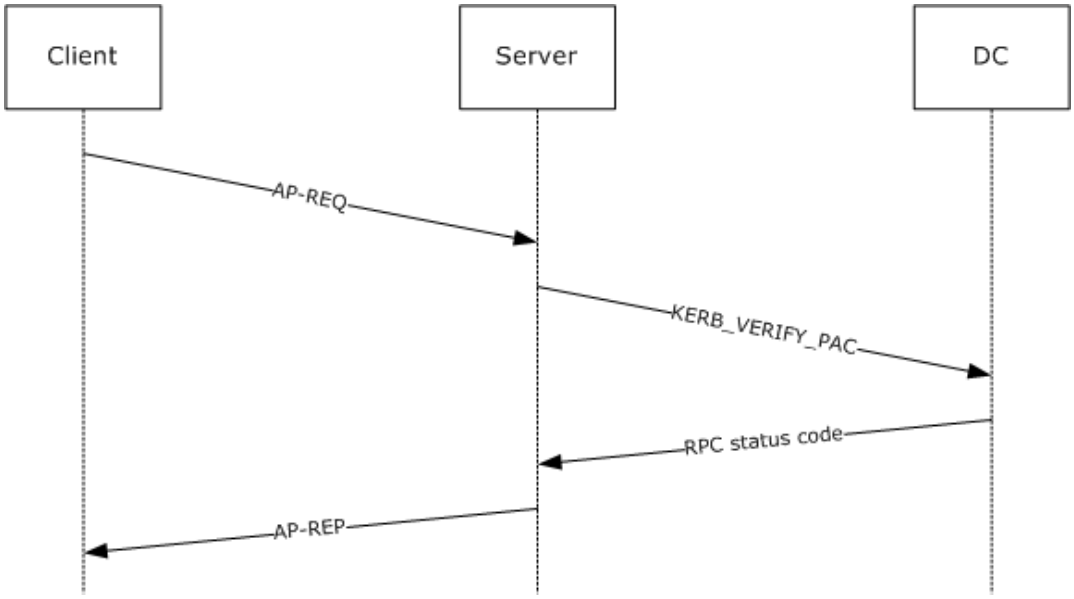


Figure 3: PAC validation

1. The client tries to access a resource requiring **Kerberos** authentication. The client sends an AP-REQ message to request authentication from the server.
2. The server passes the **PAC** to the operating system to receive an access token. The server operating system forwards the PAC signature in the AP-REQ to the **domain controller** for verification in a KERB_VERIFY_PAC message.
3. The domain controller verifies the signature on the response and returns the result to the server. The error is returned as the appropriate **RPC** status code.
4. The server verifies the AP-REQ, and sends an AP-REP if the verification is successful.

4.3 Digest Validation Protocol

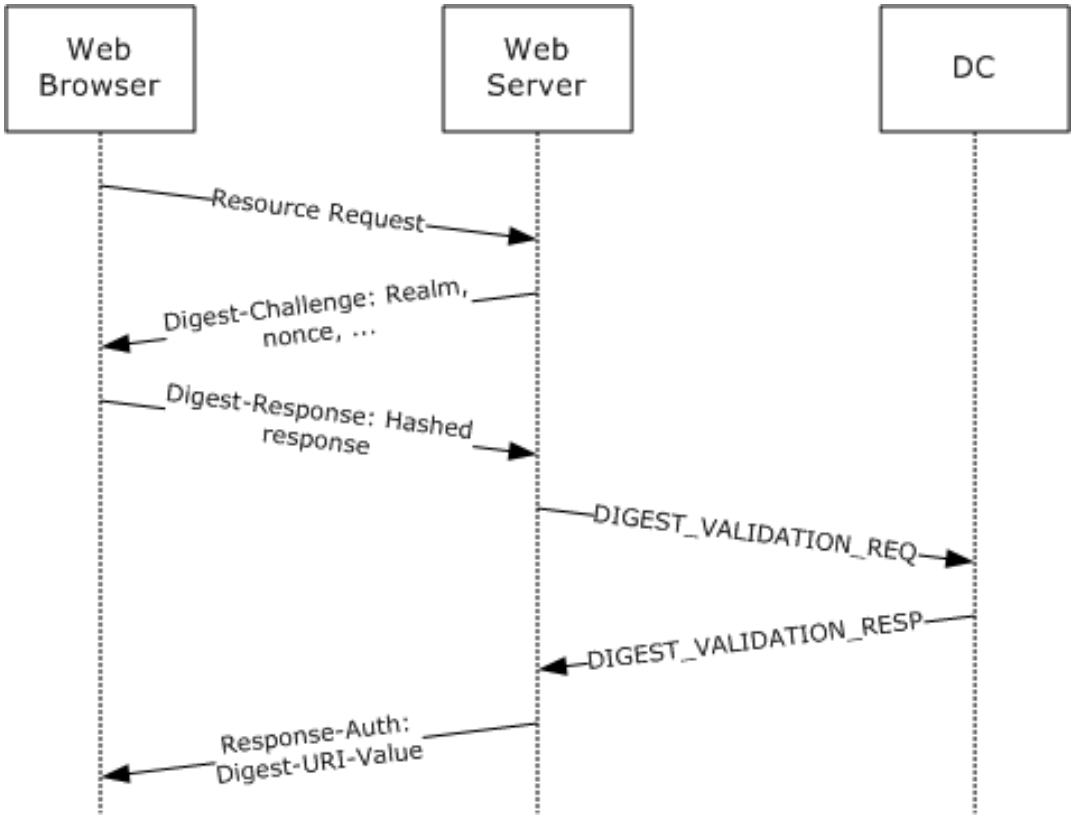


Figure 4: Digest validation protocol

1. The web server is configured to require **Digest authentication** to gain access to certain documents. A user attempts to gain access to the protected document by using a web browser. The web server returns the Digest-Challenge message, as specified in [RFC2617] section 3.3. This challenge message includes a randomly generated nonce intended to foil replay attacks.

2. The web browser obtains the user name and password for the user and constructs a response to the server's challenge. In the Digest-Response, the client proves knowledge of the user's password by performing a keyed hash over the user name, nonce, and other fields (the password is fed into the hash).

3. To validate the Digest-Response message, the web server constructs the DIGEST_VALIDATION_REQ (section 2.2.5.1) message and sends it to the **DC**. The DIGEST_VALIDATION_REQ includes the nonce and the keyed hash value from the Digest-Response message. On receiving the DIGEST_VALIDATION_REQ message, the DC validates the message by performing the following steps:

    1. Looks up the user's password by using the user name.

    2. Recomputes the keyed hash over the clear-text fields from the Digest-Response message.

    3. Compares the resulting value to the value sent by the client.

4. If the DC's computed hash and the hash sent by the client match, the DC creates and sends back the DIGEST_VALIDATION_RESP (section 2.2.5.2) message with Status indicating successful authentication (that is, STATUS_SEVERITY_SUCCESS), as specified in [MS-ERREF] section 2.3, and authorization information for the user's account (the **PAC**). Otherwise, the DC returns an

error code as an error status in NRPC API. It does not send back the DIGEST_VALIDATION_RESP message.

5. For mutual authentication, the server has the option to send a keyed hash over the URI that the client requested, and return it to the client in the Response-Auth message. Note that sending the Response-Auth message applies only to Digest authentication when used as a Simple Authentication and Security Layer (SASL) mechanism, as specified in [RFC2831] section 2.1.3.

5 Security

5.1 Security Considerations for Implementers

Authentication Protocol Domain Support does not have any built-in security mechanisms to provide authentication or to ensure confidentiality and integrity. Instead, it relies on security mechanisms that are specified in [MS-RPCE] section 5 to protect Netlogon **RPC** [MS-NRPC], which serves as the transport for the protocols defined in this document.

The **Digest authentication** protocol itself offers some level of protection (that is, it does not send the user's password in the clear) but is considered weaker than **Kerberos** [MS-KILE] and public key–based authentication (for example, client-side authentication). Consequently, the Digest Protocol Extensions are used only in environments in which these stronger mechanisms are unavailable.

5.2 Index of Security Parameters

There are no security parameters for Authentication Protocol Domain Support. All associated security parameters are specified in the Netlogon Remote Protocol [MS-NRPC], which provides all security for Authentication Protocol Domain Support messages.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

Windows Client

- Windows NT operating system
- Windows 2000 Professional operating system
- Windows XP operating system
- Windows Vista operating system
- Windows 7 operating system
- Windows 8 operating system
- Windows 8.1 operating system
- Windows 10 operating system
- Windows 11 operating system

Windows Server

- Windows 2000 Server operating system
- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system
- Windows Server 2022 operating system
- Windows Server 2025 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the

SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 2.2.1: Although the Kerberos protocol is preferred for **interactive logon**, the **NTLM** protocol is always used for interactive logon when the **domain controller** is running Windows NT Server 4.0 operating system.

<2> Section 2.2.4: The following list details differences in **PAC** validation in Windows.

- Windows 2000 Server and Windows XP do not validate the PAC when the **application server** is running under the local system context or has SeTcbPrivilege, as specified in [MS-LSAD] section 3.1.1.2.1. Otherwise, Windows 2000 Server and Windows XP use **Kerberos** PAC validation.
- Windows Server 2003 does not validate the PAC when the application server is running under the local system context, the network **service** context or has SeTcbPrivilege. Otherwise, Windows Server 2003 uses Kerberos PAC validation.
- Windows Server 2003 operating system with Service Pack 1 (SP1) and later Windows operating systems do not validate the PAC when the application server is under the local system context, the network service context, the local service context, or has SeTcbPrivilege privilege. Otherwise, Windows Server 2003 with SP1 and future service packs or later Windows operating systems use Kerberos PAC validation.
- Only Windows 2000 Server, Windows XP, and Windows Server 2003 validate the PAC by default for services. Windows still validates the PAC for processes that are not running as services. PAC validation can be enabled when the application server is not running in the context of local system, network service, or local service; or when it does not have SeTcbPrivilege, as specified in [MS-LSAD] section 3.1.1.2.1.

<3> Section 3.1.1: **NTLMServerDomainBlocked** is not supported on Windows NT, Windows 2000 Server, Windows Server 2003, or Windows Server 2008. Where supported, the default is FALSE.

<4> Section 3.1.1: These configuration values are not supported on Windows 2000 Server, Windows Server 2003, or Windows Server 2008. Where they are supported, the default value for **AccountDCBlocked** and **ResourceDCBlocked** is FALSE. The default value of **DCBlockExceptions**, where supported, is NULL.

<5> Section 3.1.1: In Windows 2000 Server, **AllowComputerLogon** is provided by the application to the **NTLM server**. This value can influence various protocol-defined flags. For example, if **AllowComputerLogon** is not set, then the **K** bit of **LogonInformation.LogonNetwork.Identity.ParameterControl** (section 3.1.5.2) is not set.

<6> Section 3.1.5: Windows does not use the NetrLogonSamLogonEx method. Windows uses NetrLogonSamLogonWithFlags (Opnum 45).

<7> Section 3.1.5: **NTLMServerDomainBlocked** is not supported on Windows NT, Windows 2000 Server, Windows Server 2003, or Windows Server 2008.

<8> Section 3.1.5: **ResourceDCBlocked** is not supported on Windows 2000 Server, Windows Server 2003, or Windows Server 2008.

<9> Section 3.1.5: **AccountDCBlocked** is not supported on Windows 2000 Server, Windows Server 2003, or Windows Server 2008.

<10> Section 3.1.5: **AllowedToAuthenticateTo** is not supported by Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012 DCs.

<11> Section 3.1.5: PROTECTED_USERS is not supported by Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012.

<12> Section 3.1.5.1: **msDS-UserAllowedToAuthenticateFrom** is not supported by Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012.

<13> Section 3.1.5.1: **msDS-ServiceAllowedToAuthenticateFrom** is not supported by Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012.

<14> Section 3.1.5.1: Windows does not use the NetrLogonSamLogonEx method. Windows uses **NetrLogonSamLogonWithFlags (Opnum 45)**.

<15> Section 3.1.5.1: In Windows NT 4.0 operating system and Windows 2000 Server, the *ValidationLevel* is NETLOGON_VALIDATION_SAM_INFO2.

<16> Section 3.1.5.1: If the DC returns a failure code, Windows fails the logon attempt. Other failure codes are also returned based on the policy (for a list of error codes, see [MS-ERREF]), and all of them result in logon failure.

When a Windows XP NETLOGON client talks with a Windows 2000 Server DC, Netlogon is responsible for negotiating the minimal *ValidationLevel* that is supported. This negotiated *ValidationLevel* is used, and the corresponding validation information is returned, as specified in [MS-NRPC] section 3.5.4.5.1. Note that the NETLOGON_VALIDATION_SAM_INFO4 structure is a superset of the NETLOGON_VALIDATION_SAM_INFO2 structure.

<17> Section 3.1.5.2: msDS-UserAllowedNTLMNetworkAuthentication and msDS-ServiceAllowedNTLMNetworkAuthentication are not supported by Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2.

<18> Section 3.1.5.2: Windows does not use the NetrLogonSamLogonEx method. Windows uses **NetrLogonSamLogonWithFlags (Opnum 45)** instead.

<19> Section 3.1.5.2: In Windows NT and Windows 2000 Server, the default *ValidationLevel* is NETLOGON_VALIDATION_SAM_INFO2.

<20> Section 3.1.5.2: In Windows 2000 Server, if **AllowComputerLogon** is not set, the **K** bit of **LogonInformation.LogonNetwork.Identity.ParameterControl** is not set. In Windows NT, NTLM servers never set the **K** bit.

<21> Section 3.1.5.2: In Windows NT, the DC cannot authenticate computer accounts.

<22> Section 3.1.5.2: When a Windows XP NETLOGON client talks with a Windows 2000 Server DC, Netlogon is responsible for negotiating the minimal ValidationLevel that is supported. This negotiated ValidationLevel is used, and the corresponding validation information is returned, as specified in [MS-NRPC] section 2.2.1.4.17. Note that the NETLOGON_VALIDATION_SAM_INFO4 structure is a superset of the NETLOGON_VALIDATION_SAM_INFO2 structure.

Except for Windows NT, the user information contained in the NETLOGON_VALIDATION_SAM_INFO4 structure is obtained by querying **Active Directory**.

<23> Section 3.1.5.2.1: On Windows 2000 operating system, subauthentication packages are installed by default.

<24> Section 3.3: The validation protocol uses the generic pass-through mechanism, as specified in [MS-NRPC] section 3.2.4.1. The **Digest validation** server is always a domain controller, and the **Digest authentication** client can be a member server of a **domain** or another DC in the same forest. The DC can also contact another DC in the same forest by using the DIGEST_VALIDATION_REQ (section 2.2.5.1) and DIGEST_VALIDATION_RESP (section 2.2.5.2) exchange, if the user's account is located in a different domain than that of the DC that receives the request. Windows clients do not support digest in this manner.

<25> Section 3.3.5.1: Windows 2000 Server can send other **AlgType** values.

<26> Section 3.3.5.2: Windows 2000 Server will not fail the DIGEST_VALIDATION_REQ request.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
2.2.2 Kerberos Ticket Validation Message Syntax	Added section to give NETLOGON_TICKET_LOGON_INFO Message usage.	Major
2.2.2.1 NETLOGON_TICKET_LOGON_INFO Message	Added section to define structure used to begin the network ticket logon flow.	Major
2.2.3 Kerberos Ticket Validation Response Message Syntax	Added section to give NETLOGON_VALIDATION_TICKET_LOGON Message usage.	Major
2.2.3.1 NETLOGON_VALIDATION_TICKET_LOGON message	Added section to define structure used to validate the logon ticket.	Major
2.2.4 Kerberos PAC Validation Message Syntax	11519 : Updated product note 2, point 3, that Windows Server 2003 with SP1 and future service packs, and later Windows operating systems do not validate the PAC, but use Kerberos PAC validation.	Major
3.2 Kerberos PAC Validation Details	Changed from KERB_VERIFY_PAC_REQUEST to NETLOGON_TICKET_LOGON_INFO message to begin ticket verification.	Major
3.2.5.1 Generating a NETLOGON_TICKET_LOGON_INFO Message	Added section to state the creation requirements.	Major
3.2.5.2 Processing a NETLOGON_TICKET_LOGON_INFO Message	Added section to state the ticket verification process.	Major
6 Appendix A: Product Behavior	Added Windows Server 2025 to the list of applicable products.	Major

8 Index