[MS-SNTP]:
Network Time Protocol (NTP) Authentication Extensions

Revision Summary

Date	Revision History	Revision Class	Comments
03/14/2007	1.0		Version 1.0 release
04/10/2007	1.1		Version 1.1 release
05/18/2007	1.2		Version 1.2 release
06/08/2007	1.2.1	Editorial	Revised and edited the technical content.
07/10/2007	1.2.2	Editorial	Revised and edited the technical content.
08/17/2007	1.2.3	Editorial	Revised and edited the technical content.
09/21/2007	1.2.4	Editorial	Revised and edited the technical content.
10/26/2007	1.2.5	Editorial	Revised and edited the technical content.
01/25/2008	1.2.6	Editorial	Revised and edited the technical content.
03/14/2008	2.0	Major	Updated and revised the technical content.
06/20/2008	2.0.1	Editorial	Revised and edited the technical content.
07/25/2008	2.1	Minor	Updated the technical content.
08/29/2008	2.1.1	Editorial	Revised and edited the technical content.
10/24/2008	2.1.2	Editorial	Revised and edited the technical content.
12/05/2008	3.0	Major	Updated and revised the technical content.
01/16/2009	4.0	Major	Updated and revised the technical content.
02/27/2009	5.0	Major	Updated and revised the technical content.
04/10/2009	6.0	Major	Updated and revised the technical content.
05/22/2009	7.0	Major	Updated and revised the technical content.
07/02/2009	8.0	Major	Updated and revised the technical content.
08/14/2009	8.1	Minor	Updated the technical content.
09/25/2009	8.2	Minor	Updated the technical content.
11/06/2009	8.3	Minor	Updated the technical content.
12/18/2009	9.0	Major	Updated and revised the technical content.
01/29/2010	10.0	Major	Updated and revised the technical content.
03/12/2010	10.0.1	Editorial	Revised and edited the technical content.
04/23/2010	11.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
06/04/2010	12.0	Major	Updated and revised the technical content.
07/16/2010	13.0	Major	Significantly changed the technical content.
08/27/2010	14.0	Major	Significantly changed the technical content.
10/08/2010	15.0	Major	Significantly changed the technical content.
11/19/2010	16.0	Major	Significantly changed the technical content.
01/07/2011	17.0	Major	Significantly changed the technical content.
02/11/2011	18.0	Major	Significantly changed the technical content.

Contents

1 Introduction

This document specifies authentication extensions to the Network Time Protocol (NTP) version 3, as specified in [RFC1305], and the Simple Network Time Protocol (SNTP) version 4, as specified in [RFC2030].

[RFC1305] Appendix C describes a mechanism similar to the authentication extensions documented here. The extensions documented here provide a strong checksum algorithm and use keying material that is readily available to systems joined to a **domain**.

SNTP, as specified in [RFC2030], provides a simplified version of NTP. [RFC2030] section 4 refers to the **Authenticator** field of [RFC1305] Appendix A; the extensions documented here replace that referral and specify the way in which an SNTP message exchange can be secured.

1.1 Glossary

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

domain
domain controller (DC)
domain controller locator
domain name (3)
domain object
fully qualified domain name (FQDN) (1)
primary domain controller (PDC)
relative identifier (RID)
security identifier (SID)
security principal (1)
time service
trusted domain object (TDO)

The following terms are specific to this document:

read-only domain controller (RODC): A **domain controller (DC)** that does not accept original updates. An **RODC** does not necessarily store full information for all objects. An **RODC**, for example, might not store passwords or password hashes for machine account objects and **TDOs**. An **RODC** cannot be the **primary domain controller (PDC)** for its **domain**. For more information, see [MS-ADTS].

writable domain controller (writable DC): Synonymous with **domain controller (DC)**, as distinct from an **RODC**.

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site,

http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", July 2006.

[MS-GPOL] Microsoft Corporation, "Group Policy: Core Protocol Specification", June 2007.

[MS-NRPC] Microsoft Corporation, "Netlogon Remote Protocol Specification", March 2007.

[MS-RRP] Microsoft Corporation, "Windows Remote Registry Protocol Specification", July 2006.

[MS-SCMR] Microsoft Corporation, "Service Control Manager Remote Protocol Specification", May 2007.

[MS-W32T] Microsoft Corporation, "W32Time Remote Protocol Specification", July 2006.

[RFC1305] Mills, D. L., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992, http://www.ietf.org/rfc/rfc1305.pdf

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt

[RFC2030] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 2030, October 1996, http://www.rfc-editor.org/rfc/rfc2030.txt

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

1.2.2 Informative References

[MD5Collision] Klima, V., "Tunnels in Hash Functions: MD5 Collisions Within a Minute", March 2006, http://eprint.iacr.org/2006/105.pdf

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MSFT-WTSRE] Microsoft Corporation, "Windows Time Service Registry Entries", http://technet.microsoft.com/en-us/library/cc773263(WS.10).aspx#w2k3tr_times_tools_uhlp

[MSWINREG] Microsoft Corporation, "Registry", http://msdn.microsoft.com/en-us/library/ms724871.aspx

1.3 Overview

1.3.1 Background

NTP version 3 [RFC1305] is used to synchronize time between a client and a server. The client sends a request for time synchronization information to the server, and the server replies with the information requested.

The NTP authentication extension in [RFC1305] Appendix C is designed to protect against unauthorized responses by using a crypto-checksum computed by the server and checked by the client. The computation that is used is a predefined encryption algorithm with cryptographic keys indexed by a key identifier included in the message. However, no provisions exist in [RFC1305] for how to securely distribute and maintain cryptographic keys or the key identifier. The extensions described in this document explain how the Microsoft Windows® implementation of the NTP protocol performs authentication utilizing the NTP packet format defined in [RFC2030] with a custom implementation of [RFC1305] Appendix C authentication, as described in section 2.2.

SNTP [RFC2030] is a simplified version of NTP that provides a coarser granularity of clock synchronization. [RFC2030] itself does not provide for authentication and refers to the relevant authentication fields of [RFC1305] Appendix C. The extensions specified in this document provide a way to add authentication to SNTP to the same degree as provided in NTP.

[RFC1305] section 3 refers to local clock in NTP. There can be implementation-specific parameters that affect the algorithms governing local clock. See [MSFT-WTSRE] for an informative description of examples of such parameters.

1.3.2 Extensions

The NTP Authentication Extensions use the Netlogon Remote Protocol (as specified in [MS-NRPC]) in a domain environment for authentication between a client and server that are capable of establishing a secure connection.

As specified in [MS-NRPC] section 3.5.5.3, the client uses the Netlogon **domain controller locator** service to find a **domain controller** that is a time source. The client and domain controller must have preestablished trusted account information in the domain. The preestablished trusted account information is used to establish cryptographic keys and associated key identifiers for NTP authentication between the client and server.

The client sends an authentication request for time-synchronization information with the key identifier. The server constructs the response with the time information requested. The server uses Netlogon message protection methods, as specified in [MS-NRPC] section 3.5.5.8, to compute the crypto-checksum of the message. The server replies in an authentication response with the time information requested and the computed crypto-checksum. The client authenticates the authentication response by computing and matching the checksum using the Netlogon message protection methods, as specified in [MS-NRPC] section 3.5.5.8. The client accepts only authenticated responses. The sequence diagram is as described in sections 3.1.5 and 3.2.5.

This document describes an authentication mechanism that differs from the interim NTP authentication extension defined in [RFC1305] Appendix C as follows:

- Uses a modified authenticator format and semantics, as described in section 2.2.
- Uses a modified algorithm to compute the checksum, as described in sections 3.1.5 and 3.2.5.
- Specifies how keys are identified by a client that is synchronizing time using the authentication extension against a domain controller, as described in sections 3.1.3, 3.1.5, and 3.1.6.
- Specifies how key identifiers must be interpreted by a server that is providing time synchronization for a client using the authentication extension, as described in section 3.2.5.
- Uses modified authentication procedures, as described in sections 3.1.5, 3.1.6, and 3.2.5.

The NTP Authentication Extensions also apply to SNTP, as specified in [RFC2030]. The extensions specified in this document override the statements in [RFC2030] section 4 pertaining to the **Authenticator** field, and instead specify how the field is to be used and interpreted.

The NTP Authentication Extensions are defined only for the following NTP and SNTP association modes (as specified in [RFC1305] sections 3.2.1 and 3.3): client, server, and symmetric active. The client/server mode refers to the roles within the context of the NTP protocol, as opposed to roles within a network system.

As well as extending [RFC1305], the NTP Authentication Extensions apply directly to SNTP, as specified in [RFC2030]. For simplicity, only the terms NTP and [RFC1305] are generally used in the main body of this document. All references to NTP and [RFC1305] must be taken to apply equally to SNTP and [RFC2030] unless the text clearly specifies otherwise.<1>

1.4 Relationship to Other Protocols

For locating a domain controller that is a time source, these extensions rely on the Netlogon domain controller locator service, as specified in [MS-NRPC] section 3.5.5.3. For authenticating message exchanges, these extensions rely on NetLogon message protection methods, as specified in [MS-NRPC] section 3.5.5.8.

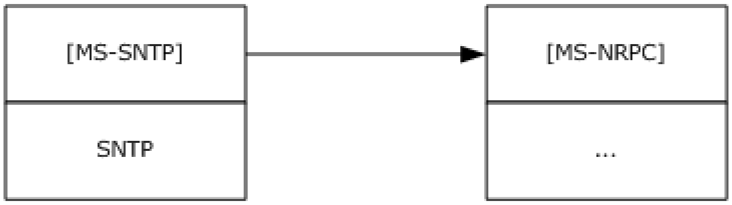

Figure 1: Protocol relationships

The W32Time Remote Protocol, as specified in [MS-W32T], is an RPC interface protocol used for controlling and monitoring the Microsoft Windows® implementation of NTP. The protocol relationship between the W32Time Remote Protocol and this protocol is illustrated in [MS-W32T] section 1.4.

1.5 Prerequisites/Preconditions

The NTP Authentication Extensions specify NTP authentication in a domain environment. The extensions, which rely on the Netlogon Remote Protocol (as specified in [MS-NRPC]), assume that a secure connection can be established between the client and a domain controller.

1.5.1 Client Configuration for Authentication

A client might need to be manually configured to participate in the NTP Authentication Extensions detailed in this specification.

On a machine that is joined to a domain, the client must use these authentication extensions to synchronize time.

1.6 Applicability Statement

The NTP Authentication Extensions are designed for authenticated time synchronization in domains between clients and domain controllers.

Clients that are joined to a domain must use the NTP Authentication Extensions to synchronize time.

1.7 Versioning and Capability Negotiation

The NTP Authentication Extensions incorporate implicit capability negotiation based on the NTP message length, as described in sections 3.1.5, 3.1.6, and 3.2.5.

NTP version 3, as specified in [RFC1305], and SNTP version 4, as specified in [RFC2030], apply to this extension.

1.8 Vendor-Extensible Fields

The NTP Authentication Extensions do not define any vendor-extensible fields. They do, however, redefine the **Authenticator** field of [RFC1305] from 96 to 160 bits (per [RFC2030]), as described in section 2.2.

1.9 Standards Assignments

The NTP Authentication Extensions do not use any additional standards assignments other than the ones in the base protocols.

2 Messages

This section describes how the **Authenticator** field is encapsulated on the wire.

2.1 Transport

All NTP messages are transported through the User Datagram Protocol (UDP) in the NTP data format, as specified in [RFC1305] Appendix A.

NTP messages do not include a message length field. The NTP message length is calculated based on the payload length in the UDP header and excludes the length of the UDP header.

Security parameters used in the NTP Authentication Extensions are key identifiers, as described in sections 3.1.6 and 3.2.5.

2.2 Message Syntax

The complete format of an NTP message conforms to [RFC2030] Section 4 and implements the optional **Authenticator** field. This document defines no fields other than the optional **Authenticator** field. For more information on NTP Version 3 message formats, see [RFC1305] Appendix A.

Note In accordance with [RFC1305], all fields are in big-endian (network byte order) format unless otherwise specified.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="2">LI</td><td colspan="3">VN</td><td colspan="3">Mode</td><td colspan="8">Stratum</td><td colspan="8">Poll Interval</td><td colspan="8">Precision</td></tr>
<tr><td colspan="32">Root Delay</td></tr>
<tr><td colspan="32">Root Dispersion</td></tr>
<tr><td colspan="32">Reference Clock Identifier</td></tr>
<tr><td colspan="32">Reference Timestamp (64 bits)</td></tr>
<tr><td colspan="32">Original Timestamp (64 bits)</td></tr>
<tr><td colspan="32">Receive Timestamp (64 bits)</td></tr>
<tr><td colspan="32">Transmit Timestamp (64 bits)</td></tr>
<tr><td colspan="32">Authenticator (optional) (160 bits)</td></tr>
</table>

The MS-SNTP protocol supports modes according to [RFC1305] section 3.4.2 (Table 5 "Modes and Actions"), except that it does not support broadcast clients, NTP control messages, or symmetric passive mode. The NTP Authentication Extensions are defined only for the following NTP and SNTP association modes (as specified in [RFC1305] sections 3.2.1 and 3.3): client, server, and symmetric active, as defined in this document.

The minimum polling interval (NTP.MINPOLL) and maximum polling interval (NTP.MAXPOLL) are defined in [RFC1305] section 3.2.7, Table 4. These intervals are not configurable using the NTP Authentication Extensions.<2>

The following is an ASCII identifier for the **Reference Clock Identifier** field in addition to those defined in [RFC2030] section 4 "NTP Message Format".

Code	Informative Meaning
VMTP	Uncalibrated virtual local clock.

If IPv6 is used as the transport mechanism, an MD5 hash [RFC1321] of the IPv6 address is calculated and the least significant 32 bits of the hash are used to set the **Reference Clock Identifier** field. By using a hash of the IPv6 address, a time source can be relatively identified in its uniqueness.

The optional **Authenticator** field used in the NTP authentication extension was originally defined in [RFC1305] Appendix C as a 96-bit field.

The **Authenticator** field defined by the NTP Authentication Extensions is a 160-bit field. When using these extensions, the total NTP message length is 68 bytes. Clients and domain controllers rely on the NTP message length to detect the use of these extensions. For more information about the **Authenticator** field, see sections 2.2.1 and 2.2.2.

The key identifier in the **Key Identifier** subfield of the **Authenticator** field identifies the cryptographic key used to generate the crypto-checksum. The association of key identifier and cryptographic key is established as a prerequisite, as explained in section 1.3.2. The method to compute the crypto-checksum is specified in sections 3.1.5 and 3.2.5.

2.2.1 Client NTP Request

The subfields of the **Authenticator** field of the Client NTP Request message include the following.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Key Identifier																															
Crypto-Checksum																															
...																															
...																															
...																															

Key Identifier (4 bytes): A 32-bit unsigned integer in little-endian byte order (least significant byte first). This field identifies the cryptographic key used to generate the crypto-checksum. The least significant 31 bits are the least significant 31 bits of a **RID** for a trusted account, as described in section 3.1.3. The most significant bit of the key identifier is a 1-bit key selector that identifies the cryptographic key to use in the pair of keys associated with the account. For more information on the semantics of this key selector, see sections 3.1.5, 3.1.6, and 3.2.5.

Crypto-Checksum (16 bytes): A 128-bit crypto-checksum that the encryption procedure computes. For more information, see section 3.2.5.<3>

2.2.2 Server NTP Response

The subfields of the **Authenticator** field of the Server NTP Response message include the following.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="32">Key Identifier</td></tr>
<tr><td colspan="32">Crypto-Checksum</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Key Identifier (4 bytes): A 32-bit unsigned integer in little-endian byte order (least significant byte first). It identifies the cryptographic key used to generate the crypto-checksum, as specified in section 2.2.1. For more information, see section 3.1.5.<4>

Crypto-Checksum (16 bytes): A 128-bit crypto-checksum computed by an encryption procedure. For more information, see section 3.2.5.

All other fields are returned according to [RFC1305].<5>

3 Protocol Details

The NTP Authentication Extensions operate between a client and a server during authenticated time synchronization.

3.1 Client Details

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.

IsLastIncomingResponseAuthenticated: A Boolean value that tracks whether the last incoming response was authenticated successfully.

Trusted Domain: A null-terminated Unicode string that represents the name of a trusted domain in which a secure connection can be established between the client and server.

Key Selector: This element is the index into the crypto-checksums that the client is currently requesting for use in authenticating responses. The crypto-checksums are computed by the Netlogon message protection method using the cryptographic keys. This element is an unsigned integer and its value MUST be either 0 or 1.

RID: An unsigned, 32-bit integer that specifies the RID that uniquely identifies a trusted account whose passwords are used for establishing the secure connection in the trusted domain.

LargePhaseOffset: An unsigned 32-bit integer that specifies the time variation from the computer's clock (phase offset) that a time sample must have to be considered a spike. Time samples that have time variations larger than the **LargePhaseOffset** value are considered spikes.

This element, expressed in 100-nanosecond (ns) units, is exposed through the Microsoft Windows® registry via the following registry value.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	LargePhaseOffset
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol as specified in [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

HoldPeriod: An unsigned 32-bit integer that specifies how many time samples larger than the **LargePhaseOffset** the client must receive in a series before subsequent time samples are accepted.

This element is exposed through the Windows registry via the following registry value:

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	HoldPeriod
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol as specified in [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

HoldCount: An unsigned 32-bit integer that tracks the number of previous samples received in a series that have been considered to be a spike.

SpikeWatchPeriod: An unsigned 32-bit integer that specifies the amount of time that suspicious time samples are received from a time source before subsequent time samples are accepted.

This element, which is expressed in seconds, is exposed through the Windows registry via the following registry value:

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	SpikeWatchPeriod
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol as specified in [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

TimeSourceType: A null-terminated Unicode string that controls how the client locates a time source. The value MUST be one of the following.

Value	Meaning
"NoSync"	No time source is used.
"NTP"	Locate a time source by using the time source defined in the **NtpServer** element.
"NT5DS"	Locate a time source by using Netlogon as defined in section 3.1.3.
"AllSync"	Locate a time source by using NetLogon as defined in section 3.1.3 or by using the time source defined in the **NtpServer** element.

This element is exposed through the Windows registry via the following registry value. The value is stored as a string exactly as shown in the Value column in the table above. For more information about the Windows registry, see [MSWINREG].

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters

Attribute	Value
Name	Type
Type	REG_SZ (string)

This element is shared with the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

NtpServer: A null-terminated Unicode string that controls the time sources used if the **TimeSourceType** element is configured to use a defined time source. Each time source MUST be in the following form.

```
<Time Source>[,<Bitwise Flag>]
```

The "Time Source" MUST be in the form of a **fully qualified domain name (FQDN)** or an IP address.

The "Bitwise Flag", if included, MUST be a bitwise OR of zero or more of the following flags.

Value	Meaning
SpecialInterval 0x01	The value of the **SpecialPollInterval** element is used as the polling interval for this time source.
UseAsFallbackOnly 0x02	Use this time source only when all other time sources have failed. No preference is given among fallback time sources when multiple time sources are configured with this option.
SymmetricActive 0x04	Use the symmetric active mode when communicating with this time source.
Client 0x08	Use the client mode when communicating with this time source.

Multiple time sources are delineated by a space. For two time sources, the following form would be used.

```
<Time Source #1>[,<Bitwise Flags #1>] <Time Source #2>[,<Bitwise Flags #2>]
```

Note that "Bitwise Flags #1" is applied to "Time Source #1", and "Bitwise Flags #2" is applied to "Time Source #2". Each time source that is listed MUST be unique. There are no uniqueness constraints on how many times a given "Bitwise Flag" can appear in a list of time sources. If the bitwise flags are not provided for a time source, the bitwise flags for that time source are considered to be set to 0x0.

If both SymmetricActive and Client flags are set for a time source, Client mode will be used to communicate with that time source.

If neither SymmetricActive nor Client flags are set for a time source and the client itself is a time source, SymmetricActive mode is used to communicate with the time source.

If neither SymmetricActive nor Client flags are set for a time source and the client itself is not a time source, Client mode will be used to communicate with that time source.

The **NtpServer** element is exposed through the Windows registry via the following registry value.<6>

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters
Name	NtpServer
Type	REG_SZ (string)

This element is shared with the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

SpecialPollInterval: An unsigned, 32-bit integer that specifies the number of seconds to use as a polling interval. The applicability of the **SpecialPollInterval** element depends on certain conditions that are created by the **TimeSourceType** and **NtpServer** elements, as described earlier in this section and in section 3.1.3.

This element is exposed through the Windows registry via the following registry value.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient
Name	SpecialPollInterval
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

ResolvePeerBackoffMinutes: An unsigned integer that indicates the initial time interval, in minutes, to wait after a failure before starting a new time source selection process.

This element is exposed through the Windows registry via the following registry value.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient
Name	ResolvePeerBackoffMinutes
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

ResolvePeerBackoffMaxTimes: An unsigned integer that indicates the maximum number of times to double the wait-time interval when repeated attempts to select a time source fail.

This element is exposed through the Windows registry via the following registry value.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient
Name	ResolvePeerBackoffMaxTimes
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

AnnounceFlags: Shared with the server role in this document (section 3.2.1).

CrossSiteSyncFlags: An unsigned, 32-bit integer that specifies whether cross-site time synchronization is allowed.

The value of the key MUST be equal to one of the following values.

Value	Meaning
"None" 0x00000000	Cross-site time synchronization is not allowed.
"PdcOnly" 0x00000001	Cross-site time synchronization is allowed if the machine is a **primary domain controller (PDC)**.
"All" 0x00000002	Cross-site time synchronization is allowed.

This element is exposed through the Windows registry via the following registry value.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient
Name	CrossSiteSyncFlags
Type	REG_DWORD

This element can be set by using the Remote Registry Protocol [MS-RRP].

3.1.2 Timers

3.1.2.1 Client Polling Timer

The client polling timer (peer.peerpoll) defined in [RFC1305] section 3.2.3 is used to control the polling interval between transmitted Client NTP Request messages. The polling interval for this timer varies in a dynamic range between a minimum polling interval and a maximum polling interval, as defined in [RFC1305] section 3.4.2.<7>

The client will perform a polling interval update as specified in [RFC1305] section 3.4.9. (Poll-Update Procedure) and in the previous paragraph, except in the following cases and in the following order:

- If a special polling interval is being used per section 3.1.1, the polling interval remains unchanged.
- If a client mode or symmetric active mode is used, the minimum polling interval is used.

3.1.2.2 Domain Controller Time Source Location Retry Timer

The domain controller time source location retry timer is used to control the wait time before a retry attempt to locate a domain controller time source after a failure. Initially, the timer interval SHOULD be set to **ResolvePeerBackoffMinutes** minutes. Every time a consecutive retry happens, the time to wait is doubled, up to **ResolvePeerBackoffMaxTimes** times. If consecutive retries continue to happen after **ResolvePeerBackoffMaxTimes** is reached, the time to wait is not doubled.

3.1.3 Initialization

3.1.3.1 Initialization Steps

The client initialization begins with the procedures described in [RFC1305] section 3.4.7.1, Initialization Procedure. It is followed by the initialization steps specified here.

1. The **LargePhaseOffset**, **HoldPeriod**, **SpikeWatchPeriod**, **TimeSourceType**, **NtpServer**, **SpecialPollInterval**, **ResolvePeerBackoffMinutes**, and **ResolvePeerBackoffMaxTimes** elements SHOULD be initialized to the following values unless locally configured otherwise.<8>

   LargePhaseOffset: 50000000 (5 seconds)

   HoldPeriod: 5

   SpikeWatchPeriod: 900 (15 minutes)

   SpecialPollInterval: 3600 (1 hour)

   ResolvePeerMaxBackoffMinutes: 15

   ResolvePeerMaxBackoffMaxTimes: 7

2. If the **TimeSourceType** element is set to "NoSync", the client MUST NOT select a time source, and protocol initialization is completed. No request will be sent and the client will remain unsynchronized.

3. If the **TimeSourceType** element is set to "NTP", the client MUST select the first reachable time source from the **NtpServer** Abstract Data Model element. If the **NtpServer** element does not contain any time sources, the behavior is not constrained by protocol.

4. If the **TimeSourceType** element is set to "NT5DS", the client MUST select a time source by using the Domain Time Source Selection Algorithm described in section 3.1.3.2.

   If a time source is returned, the client MUST execute the steps in section 3.1.9.

   If a time source cannot be found, the client MUST wait for **ResolvePeerBackoffMinutes** minutes and retry step 4. Every time a consecutive retry happens, the time to wait is doubled, up to **ResolvePeerBackoffMaxTimes** times. If consecutive retries continue to happen after **ResolvePeerBackoffMaxTimes** is reached, the time to wait is not doubled.

5. If the **TimeSourceType** element is set to "AllSync", the client MUST attempt to locate a time source by using the Domain Time Source Selection Algorithm described in section 3.1.3.2.

   If a time source is returned, the client MUST execute the steps in section 3.1.9.

   If a domain controller time source cannot be found, the client MUST then select a time source from the **NtpServer** element. The client SHOULD set up a domain controller time source location retry timer to retry locating a domain controller time source.

6. The client MUST initialize the client polling timer according to [RFC1305] section 3.4.7, regardless of which time source is selected.

7. The client MAY set the **sys.refid** element to "VMTP".<9> If the **sys.refid** element is set to "VMTP", then:

   - The **sys.stratum** element MUST be set to 2.

   The **sys.refid** and **sys.stratum** elements are defined in [RFC1305] section 3.2.1, "Common Variables".

8. If the **sys.refid** element is set to "LOCL" then:

   - The **AnnounceFlags** element MUST have either the Reliable_Timeserv_Announce_Yes flag set or the Reliable_Timeserv_Announce_Auto flag set.

   The **sys.refid** element is defined in [RFC1305] section 3.2.1, "Common Variables", and the **sys.refid** value "LOCL" is defined in [RFC2030] section 4 "NTP Message Format".

3.1.3.2 Domain Time Source Selection Algorithm

The client MUST first obtain its current site by calling the **DsrGetSiteName** method described in [MS-NRPC] section 3.5.5.3.6.

The client MUST try to obtain a time source by making the calls in the following order with the specified parameters. If the client obtains at least one time source in any of the following steps, the client MUST NOT make subsequent calls.

1. To obtain a reliable time source in the parent domain in the current site, the client MUST call the **DsrGetDcNameEx2** method described in [MS-NRPC] section 3.5.5.3.1 with the following parameters.

   DomainName: MUST be set to the **domain name** of the parent domain, which is obtained by calling the **NetrLogonGetTimeServiceParentDomain** method in [MS-NRPC] section 3.5.5.8.6.

   SiteName: If the **CrossSiteSyncFlags** abstract data model element is set to ALL, the client MUST set this parameter to the site obtained by using **DsrGetSiteName**; otherwise, this parameter MUST be set to NULL.

Flags: The "I" and "K" bits described in [MS-NRPC] section 3.5.5.3.3 MUST be set.

The client MUST validate that the **Flags** field in the returned **DomainControllerInfo** structure has bits G and I set (as described in [MS-NRPC] section 2.2.1.2.1); otherwise, proceed to the next step.

2. To obtain a reliable time source in the current domain in the current site, the client MUST call **DsrGetDcNameEx2** with the following parameters.

   DomainName: This parameter MUST be set to the domain name.

   SiteName: If **CrossSiteSyncFlags** is set to ALL, the client MUST set this parameter to the site obtained by using **DsrGetSiteName**; otherwise, this parameter MUST be set to NULL.

   Flags: The "I" and "K" bits described in [MS-NRPC] section 3.5.5.3.3 MUST be set.

   The client MUST validate that the **Flags** field in the returned **DomainControllerInfo** structure has bits G and I set (as described in [MS-NRPC] section 2.2.1.2.1); otherwise, proceed to the next step.

3. To obtain a PDC in the current domain in the current site, the client MUST call **DsrGetDcNameEx2** with the following parameters.

   DomainName: this parameter MUST be set to the domain name.

   SiteName: If **CrossSiteSyncFlags** is set to ALL, the client MUST set this parameter to the site obtained by using **DsrGetSiteName**; otherwise, this parameter MUST be set to NULL.

   Flags: The "I" and "E" bits described in [MS-NRPC] section 3.5.5.3.3 MUST be set.

   The client MUST validate that the **Flags** field in the returned **DomainControllerInfo** structure has bits A and G set (as described in [MS-NRPC] section 2.2.1.2.1); otherwise, proceed to the next step.

4. To obtain a reliable time source in the parent domain in any site, the client MUST call **DsrGetDcNameEx2** with the following parameters.

   DomainName: MUST be set to the domain name of the parent domain, which is obtained by calling the **NetrLogonGetTimeServiceParentDomain** method in [MS-NRPC] section 3.5.5.8.6.

   SiteName: This parameter MUST be set to NULL.

   Flags: The "I" and "K" bits described in [MS-NRPC] section 3.5.5.3.3 MUST be set.

   The client MUST validate that the **Flags** field in the returned **DomainControllerInfo** structure has bit I set (as described in [MS-NRPC] section 2.2.1.2.1); otherwise, proceed to the next step.

5. To obtain a reliable time source in the current domain in any site, the client MUST call **DsrGetDcNameEx2** with the following parameters.

   DomainName: this parameter MUST be set to the domain name.

   SiteName: This parameter MUST be set to NULL.

   Flags: The "I" and "K" bits described in [MS-NRPC] section 3.5.5.3.3 MUST be set.

   The client MUST validate that the **Flags** field in the returned **DomainControllerInfo** structure has bit I set (as described in [MS-NRPC] section 2.2.1.2.1); otherwise, proceed to the next step.

6. To obtain a PDC in the current domain in any site, the client MUST call **DsrGetDcNameEx2** with the following parameters.

   DomainName: this parameter MUST be set to the domain name.

   SiteName: This parameter MUST be set to NULL.

   Flags: The "I" and "E" bits described in [MS-NRPC] section 3.5.5.3.3 MUST be set.

   The client MUST validate that the **Flags** field in the returned **DomainControllerInfo** structure has bit A set (as described in [MS-NRPC] section 2.2.1.2.1).

If a validated time source was obtained in one of the above steps, it is returned as the result of this algorithm. If no time source was obtained, none is returned.

3.1.4 Higher-Layer Triggered Events

A higher-layer triggered event is used to immediately start the time synchronization process. This event is triggered by the W32TimeSync method in [MS-W32T]. For example, when an end user wants to immediately initiate synchronization of a client computer's time with that of a server, and when the client computer is configured for authenticated NTP time synchronization, the W32TimeSync method is used to trigger this event. When this event is triggered, the client polling timer specified in section 3.1.2 expires immediately.

The client also responds to two higher-layer triggered events related to domain join and domain unjoin. These higher-layer triggered events are used to modify the configuration of the client when entering or leaving membership in a domain, respectively. For more information on domain membership, refer to [MS-DISO] section 4.7.1.2.

3.1.4.1 Domain Join Processing

The client MUST set the **TimeSourceType** abstract data model element (section 3.1.1) to the value "NT5DS". The client MUST then reinitialize its configuration accordingly, as indicated in section 3.1.3.

3.1.4.2 Domain Unjoin Processing

The client MUST set the **TimeSourceType** abstract data model element (section 3.1.1) to the value "NTP". The client MUST then reinitialize its configuration accordingly, as indicated in section 3.1.3.

3.1.4.3 Group Policy Update Notification

When the Policy Application Event described in [MS-GPOL] section 3.2.1.24 is raised, the configuration MUST re-initialize. See section 3.1.3.

3.1.4.4 Service Control Manager Update Notification

When the SERVICE_CONTROL_PARAMCHANGE event described in [MS-SCMR] is triggered, the configuration MUST be re-initialized accordingly, as indicated in section 3.1.3.

3.1.4.5 Network Change Notification

When the IP address changes, the configuration MUST re-initialize. See section 3.1.3.

3.1.5 Message Processing Events and Sequencing Rules

The MS-SNTP protocol supports modes according to [RFC1305] section 3.4.2 (Table 5 "Modes and Actions"), except that it does not support broadcast clients, NTP control message, or symmetric passive mode. If the client receives a message with an unsupported mode, the message MUST be ignored as specified in [RFC1305] section 3.4.3. The NTP Authentication Extensions are defined only for the following NTP and SNTP association modes (as specified in [RFC1305] sections 3.2.1 and 3.3): client, server, and symmetric active.

The client prepares a Client NTP Request message header in the same way it occurs in the base Network Time Protocol (NTP), which is described in [RFC1305] section 3.4.2.

3.1.5.1 Authentication Request Behavior

The following diagram illustrates the client logic for generating a Client NTP Request message with authentication.

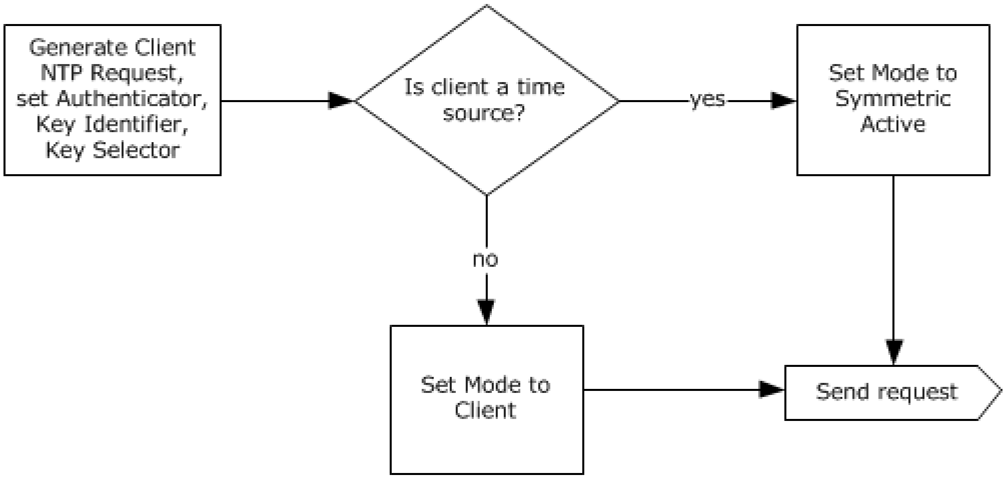
Generate Client NTP Request, set Authenticator, Key Identifier, Key Selector
Is client a time source?
yes
Set Mode to Symmetric Active
no
Set Mode to Client
Send request

Figure 2: Authentication request generation

When a client requests authentication, the Client NTP Request message length is 68 bytes. The client sets the **Authenticator** field of the Client NTP Request message as described in section 2.2.1, writing the least significant 31 bits of the RID value into the least significant 31 bits of the **Key Identifier** subfield of the authenticator, and then writing the **Key Selector** value into the most significant bit of the **Key Identifier** subfield.<10>

The client SHOULD set the **Mode** field of the request to Symmetric Active if the client is a time source. The syntax and semantics for the **Mode** field of the Client NTP Request message are specified in [RFC1305] Appendix A.<11>

The client sends the Client NTP Request message to the server as it does in the base Network Time Protocol, which is described in [RFC1305] section 3.4.2.

The following diagram illustrates the client logic for processing a Server NTP Response message received in response to a Client NTP Request message that requested authentication.<12>

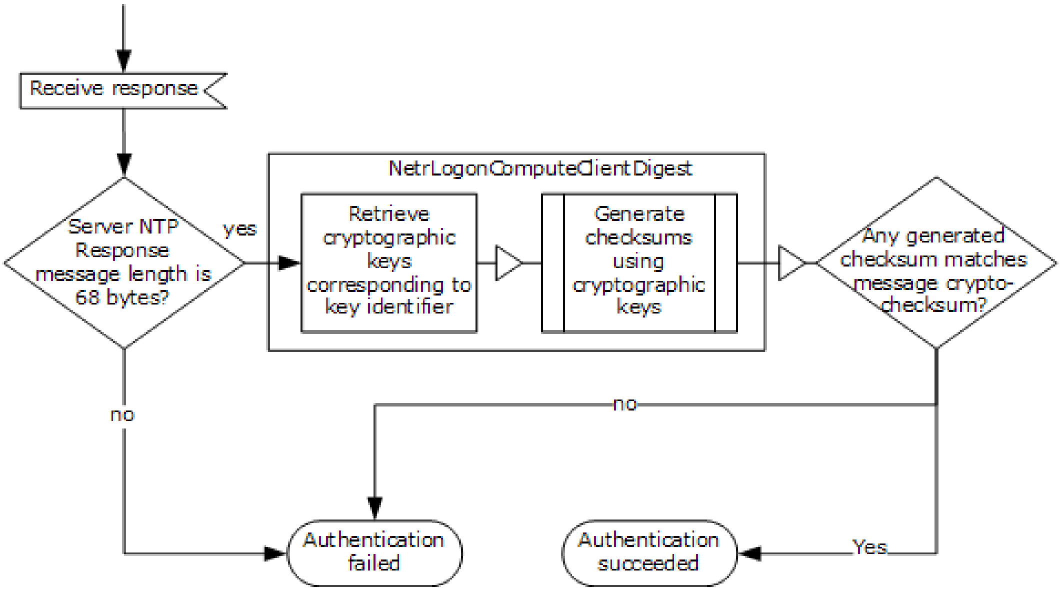
Receive response
Server NTP Response message length is 68 bytes?
yes
no
NetrLogonComputeClientDigest
Retrieve cryptographic keys corresponding to key identifier
Generate checksums using cryptographic keys
Any generated checksum matches message crypto-checksum?
no
Yes
Authentication failed
Authentication succeeded

Figure 3: Authentication response processing

The Server NTP Response message length MUST be 68 bytes. If the message length does not meet this requirement, the authentication fails.

The client MUST ignore the **Key Identifier** subfield of the Server NTP Response message (as should any recipient of the Server NTP Response message).

The client uses the **NetrLogonComputeClientDigest** method (as specified in [MS-NRPC] section 3.5.5.8.3) to compute crypto-checksums for the first 48 bytes of the Server NTP Response message, with the following input parameters:

- *ServerName* MUST be set to NULL.
- *DomainName* MUST be set to the value of the **Trusted Domain** element.
- *Message* MUST refer to the first 48 bytes of the Server NTP Response message.
- *MessageSize* MUST be set to 48.

The **NetrLogonComputeClientDigest** method computes two crypto-checksums using the pair of passwords associated with the trusted account.

The client compares each computed crypto-checksum with the **Crypto-Checksum** subfield in the Server NTP Response message. If the **Crypto-Checksum** subfield matches any of the computed crypto-checksums, the authentication succeeds. Otherwise, the authentication fails. A client MUST compare all computed crypto-checksums before determining that the authentication has failed; however, it SHOULD NOT continue to compare crypto-checksums after it has determined that at least one of its computed crypto-checksums matches the **Crypto-Checksum** subfield.

If authentication succeeds, the client continues processing the response to synchronize time the same way it occurs in the base NTP protocol, which is described in [RFC1305] section 3.4.3.

If authentication fails, the response MUST be ignored, and the client MUST NOT perform time synchronization using the response.

The following state diagram illustrates updates to the client's state elements based on success or failure of message authentication.

Notes on the following state diagram: In authenticated NTP, all state transitions are triggered by timer expiry. On expiration of the client polling timer, an authenticated NTP client attempts an authenticated NTP exchange with the NTP server. Based on the success or failure of that attempt, it updates state elements and transitions to the next state. The labels on the following arcs indicate the trigger of authentication success or failure that causes transition to the next state. For each trigger of authentication success or failure, there is an implicit trigger of "Timer Expiry" because it is the expiration of the polling timer that causes an authentication attempt. The "Timer Expiry" label has been omitted from the following arcs for clarity. Also note that the state element assignments in the state boxes are carried out upon entry into the state, not on exit.

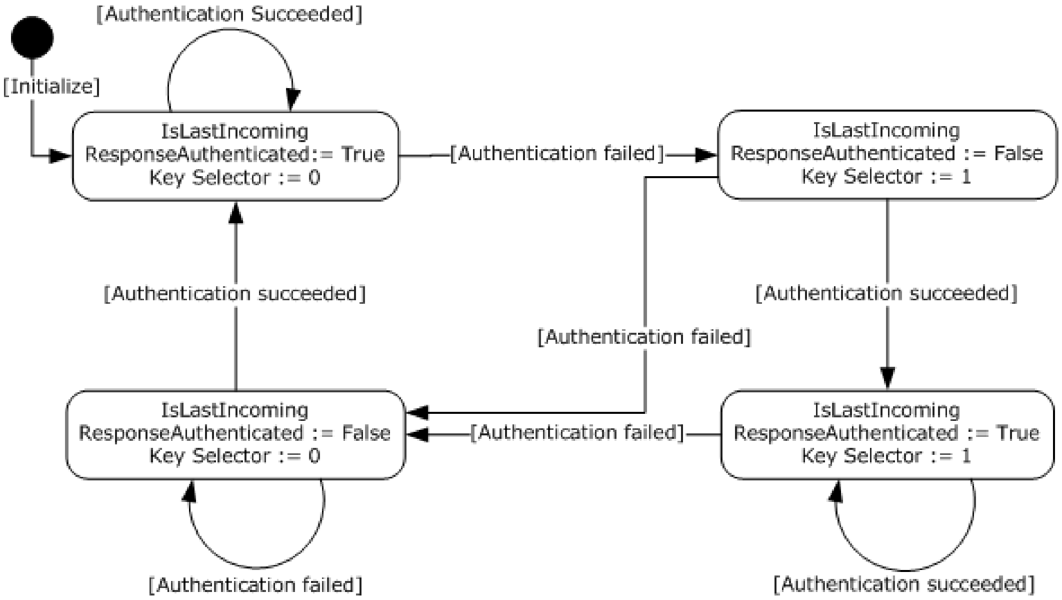
[Authentication Succeeded]
[Initialize]
IsLastIncoming ResponseAuthenticated:= True Key Selector := 0
[Authentication failed]
IsLastIncoming ResponseAuthenticated := False Key Selector := 1
[Authentication succeeded]
[Authentication failed]
[Authentication succeeded]
IsLastIncoming ResponseAuthenticated := False Key Selector := 0
[Authentication failed]
IsLastIncoming ResponseAuthenticated := True Key Selector := 1
[Authentication failed]
[Authentication succeeded]

Figure 4: Client element updates

3.1.5.2 Special Root Dispersion Behavior

The client SHOULD set the **Root Dispersion** value in the request to 0xAAAAAAAA. If the response that is returned from the server contains a **Root Dispersion** value of 0xAAAAAAAA, then the server's response contains a **Reference Timestamp** value that is relative to the client's clock, and the client MUST NOT process Test 6 defined in [RFC1305] section 3.4.4. If the response that is returned from the server does not contain a **Root Dispersion** value of 0xAAAAAAAA, then the server's response contains a **Reference Timestamp** value that is relative to the server's clock, and the client SHOULD process Test 6 as defined in [RFC1305] section 3.4.4.<13><14>

3.1.5.3 Test 7 Behavior

The client MUST NOT perform the portion of Test 7 that verifies that pkt.stratum <= sys.stratum as indicated in [RFC1305] section 3.4.4.

3.1.5.4 Spike Watch Mechanism

The following diagram illustrates the client logic for the evaluation and handling of spikes.

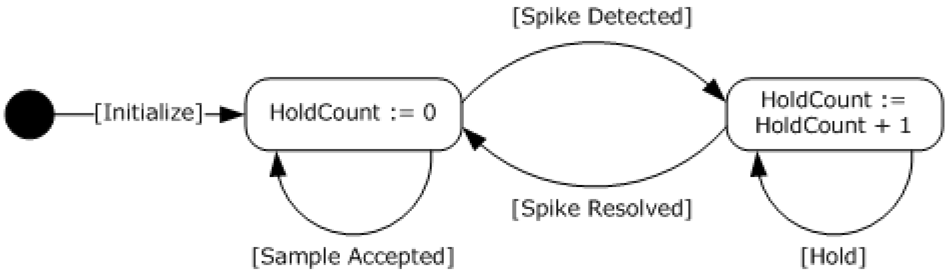

Figure 5: Client spike evaluation and processing

During initialization, the client sets the **HoldCount** element to zero.

Upon receiving a response, the client calculates the offset from the server (peer.offset) using the data contained in the response as defined in [RFC1305] section 3.4.4. If the absolute value of the calculated offset is less than the value of the **LargePhaseOffset** element, the sample is accepted and processing continues as is indicated by the [Sample Accepted] transition in the preceding diagram.

If the absolute value of the calculated offset is equal to or larger than the value of the **LargePhaseOffset** element and the client is not already in the hold state, the client will enter a hold state as indicated by the [Spike Detected] transition. In this transition, the value of **HoldCount** will be set to one and the sample will be discarded.

If the absolute value of the calculated offset is equal to or larger than the value of the **LargePhaseOffset** element and the client is already in the hold state, the client will remain in the hold state as indicated by the [Hold] transition. In this transition, the value of the **HoldCount** element will be incremented by one and the sample will be discarded.

The client will exit the hold state when one of the following three conditions has been met.

1. The value of the **HoldCount** element is equal to or greater than the value of the **HoldPeriod** element.
2. The amount of time that has elapsed since the first spike was received is equal to or greater than the value of the **SpikeWatchPeriod** element.
3. A sample is received and the absolute value of the calculated offset of that sample is less than the value of the **LargePhaseOffset** element.

Upon exiting the hold state, the **HoldCount** will be set to zero, the sample will be accepted, and processing will continue as indicated by the [Spike Resolved] transition.

3.1.6 Timer Events

3.1.6.1 Client Polling Timer Event

When the client polling timer expires, the client prepares a Client NTP Request message as described in section 3.1.5.

3.1.6.2 Domain Controller Time Source Location Retry Timer Event

When the domain controller time source location retry timer expires, the client MUST attempt to locate a time source by using the **DsrGetDcName** method defined in [MS-NRPC] section 3.5.5.3.3.

When calling the **DsrGetDcName** method, the client MUST set the "K" bit in the *Flags* parameter to indicate that a domain controller that is running a reliable **time service** is to be returned, as described in [MS-NRPC] section 3.5.5.3.3.

If a time source is found, the client MUST execute the steps in section 3.1.9 and cancel the timer.

3.1.7 Other Local Events

No additional events are used.

3.1.8 Clock Selection

Clock selection occurs as defined in [RFC1305] section 4.2, with the exception of the following scenario:

- If **sys.refid** is set to "VMTP", then **sys.peer** is set to NULL. This is done regardless of what value the **peer.stratum** element is set to.

3.1.9 Domain Controller Time Source Initialization

The client initializes the **Trusted Domain** abstract data model element by calling the **NetrLogonGetTimeServiceParentDomain** method (as specified in [MS-NRPC] section 3.5.5.8.6) with the following input parameters:

- *ServerName* MUST be set to NULL.

Upon successful completion, the output parameter *DomainName* will contain the name of a trusted domain in which a secure connection can be established between the client and server.

The client initializes the **RID** element by calling the **NetrLogonGetTrustRid** method (as specified in [MS-NRPC] section 3.5.5.8.1) with the following input parameters:

- *ServerName* MUST be set to NULL.
- *DomainName* MUST be set to the value of the **Trusted Domain** element.

Upon successful completion, the output parameter *Rid* will contain the RID that uniquely identifies a trusted account.

The client initializes the **IsLastIncomingResponseAuthenticated** abstract data model element to true and initializes the **Key Selector** abstract data model element to zero.

3.2 Server Details

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.

AnnounceFlags: A 32-bit set of bit flags that controls which NetLogon Service Bits are set, as specified in section 3.2.3.1. The value MUST be a bitwise OR of zero or more of the following flags.<15><16>

Value	Informative Meaning
Timeserv_Announce_No 0x00	Unset all the Service Bits.
Timeserv_Announce_Yes 0x01	This flag indicates to clients that this computer is available to respond to NTP Authentication Extensions time synchronization requests. Note that this setting does not indicate that time samples returned from this computer will be usable, only that this computer should be considered as a usable time source for the purposes of time source discovery.
Timeserv_Announce_Auto 0x02	This flag indicates to clients that this computer's clock is synchronized with another time source.
Reliable_Timeserv_Announce_Yes 0x04	This flag indicates that the computer's clock is synchronized with a time source operating at stratum 0 or stratum 1. This flag indicates to clients that this computer is a more reliable time source than time sources that do not have this setting.
Reliable_Timeserv_Announce_Auto 0x08	This flag indicates to clients that this computer is a more reliable time source than time sources that do not have this setting.

This element is exposed through the Microsoft Windows® registry via the following registry value. The value is stored as the hexadecimal value shown in the Value column of the preceding table.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	AnnounceFlags
Type	REG_DWORD

The setting referred to as "reliable" is the exposure of the **AnnounceFlags** Abstract Data Model element.

This element is shared with the client role in this document and the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

LocalClockDispersion: An unsigned 32-bit integer that is analogous to the Root Dispersion (sys.rootdispersion) variable defined in [RFC1305] section 3.2.1.

This element is exposed through the Windows registry via the following registry value. The value is expressed in seconds.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	LocalClockDispersion
Type	REG_DWORD

This element is shared with the server role of the W32Time Remote Protocol [MS-W32T].

This element can be set by using the Remote Registry Protocol [MS-RRP].

ResponseMode: An unsigned 32-bit integer that controls whether to respond to synchronization requests. This value MUST be one of the following.<17>

Value	Meaning
Response_Mode_DC_Only 0x00	Do not respond to synchronization requests unless the server is also a domain controller.
Response_Mode_Yes 0x01	Always respond to synchronization requests.

This element is exposed through the Windows registry via the following registry value. The value is stored as the hexadecimal value shown in the Value column of the preceding table.

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	LocalNTP
Type	REG_DWORD

This element can be set by using the Remote Registry Protocol [MS-RRP].

3.2.2 Timers

No new timers are required beyond those in the base NTP protocol, as specified in [RFC1305] section 3.2.3.

3.2.3 Initialization

The server initialization begins with the procedures described in [RFC1305] section 3.4.7.1, "Initialization Procedure". It is followed by the following initialization steps:

1. The server MAY set the **sys.refid** element to "VMTP" <18> If the **sys.refid** element is set to "VMTP" then:

   - The **sys.stratum** element MUST be set to 2.

2. If the **sys.refid** element is set to "LOCL" as defined in [RFC2030] section 4 then:

   - If the **AnnounceFlags** element is set to Reliable_Timeserv_Announce_Yes, then the sys.stratum element MUST be set to 1.

   - If the **AnnounceFlags** element is set to Reliable_Timeserv_Announce_Auto and the **Trusted Domain** element is NULL, then the server MUST execute the DsRolerGetPrimaryDomainInformation method defined in [MS-DSSP] section 3.2.5.1, using an InfoLevel value of DsRolePrimaryDomainInfoBasic. If the **MachineRole** field of the DSROLER_PRIMARY_DOMAIN_INFO_BASIC structure that is returned has a value of DsRole_RolePrimaryDomainController, then the sys.stratum element MUST be set to 1.

The **sys.refid** and **sys.stratum** elements are defined in [RFC1305] section 3.2.1, "Common Variables".

All abstract data model elements are configured by the server either during initialization or when it is informed of a configuration change through Service Control Manager (section 3.2.4).

3.2.3.1 Initializing NetLogon Service Bits

Once initialization is complete, the appropriate Netlogon Service Bits MUST be set as described in [MS-NRPC] section 3.5.5.8.5.<19>

- If the Timeserv_Announce_Yes flag in the **AnnounceFlags** abstract data model element is set, then set the "Time service is running" bit to 1.
- If the Timeserv_Announce_Auto flag in the **AnnounceFlags** abstract data model element is set, then set the "Time service is running" bit to 1 only if there is an active association with a peer.
- If the Reliable_Timeserv_Announce_Yes flag in the **AnnounceFlags** abstract data model element is set, then set the "Time service with clock hardware is running" bit to 1.
- If the Reliable_Timeserv_Announce_Auto flag in the **AnnounceFlags** abstract data model element is set, then set the "Time service with clock hardware is running" bit to 1 only if there is an active association with a peer.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Group Policy Update Notification

When the Policy Application Event described in [MS-GPOL] section 3.2.1.24 is raised, the configuration MUST re-initialize. See section 3.2.3.

3.2.4.2 Service Control Manager Update Notification

When the SERVICE_CONTROL_PARAMCHANGE event described in [MS-SCMR] is triggered, the configuration MUST be re-initialized accordingly, as indicated in section 3.2.3.

3.2.4.3 Network Change Notification

When the IP address changes, the configuration MUST re-initialize. See section 3.2.3.

3.2.5 Message Processing Events and Sequencing Rules

This section documents deviations in the behavior of the NTP Authentication Extensions from the processing rules specified in [RFC1305]. Unless otherwise stated, all other processing events and sequencing rules conform to [RFC1305].

3.2.5.1 Authentication Response Behavior

When the server receives the Client NTP Request message from the client, the server examines the NTP message length. The **Authenticator** field that the NTP Authentication Extensions specify is present if the NTP message length is 68 bytes. In this case, the server MUST respond with an authenticated NTP response. If the NTP message length is 48 bytes, the message is processed as specified in [RFC1305] section 3.4.3.<20> NTP messages of other lengths are ignored. As a result of this, the authentication mechanism defined in [RFC1305] Appendix C.1 is not supported.

The MS-SNTP protocol supports modes according to [RFC1305] section 3.4.2 (Table 5 "Modes and Actions"), except that it does not support broadcast clients, NTP control message, or symmetric passive mode. If the server receives a message with an unsupported mode, the message MUST be ignored as specified in [RFC1305] section 3.4.3. The NTP Authentication Extensions are defined only for the following NTP and SNTP association modes (as specified in [RFC1305] sections 3.2.1 and 3.3): client, server, and symmetric active.

3.2.5.1.1 Writable Domain Controller Processing of the Authenticated Server NTP Response

If the server is a **writable domain controller (writable DC)**, the server processes the authenticated Server NTP Response as follows.

The following diagram illustrates the server logic for providing authentication when responding to a Client NTP Request message using these extensions. Other NTP processing is specified in [RFC1305] section 3.4.3.<21>

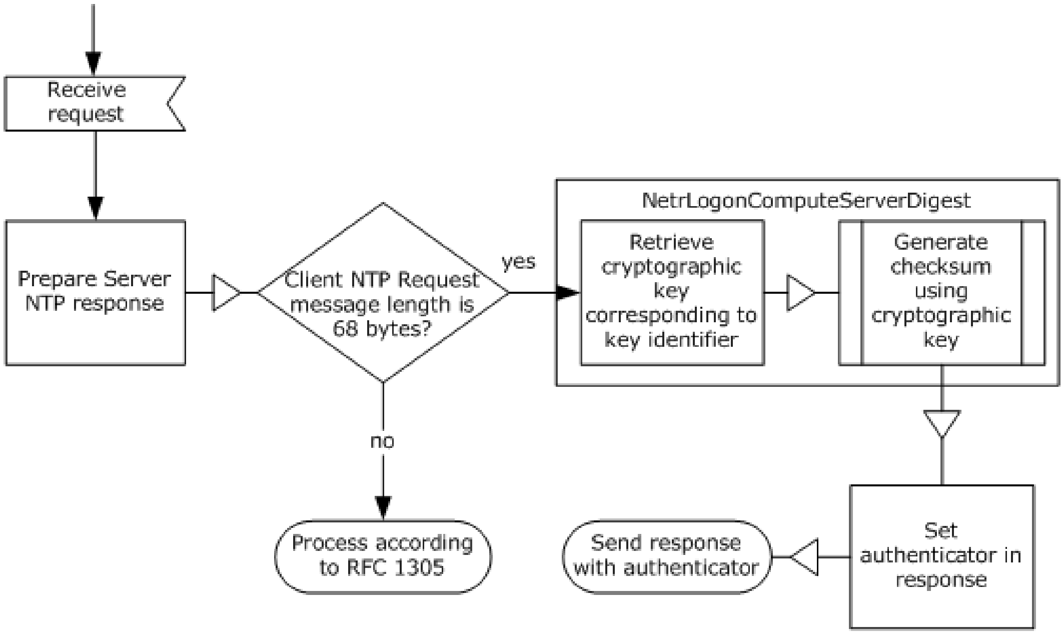
Receive request
Prepare Server NTP response
Client NTP Request message length is 68 bytes?
yes
no
NetrLogonComputeServerDigest
Retrieve cryptographic key corresponding to key identifier
Generate checksum using cryptographic key
Set authenticator in response
Send response with authenticator
Process according to RFC 1305

Figure 6: Authentication request processing

The server prepares the Server NTP Response message in the same way as for the base protocol, which is described in [RFC1305] section 3.4.3. Afterward, the server follows the server-encryption procedure.

The server MUST ignore the **Crypto-Checksum** subfield of the Client NTP Request message (as should any recipient of the Client NTP Request message).

The server retrieves the RID from the least significant 31 bits of the **Key Identifier** subfield of the **Authenticator** field of the Client NTP Request message. The server uses the **NetrLogonComputeServerDigest** method (as specified in [MS-NRPC] section 3.5.5.8.2) to compute crypto-checksums with the following input parameters:

- *ServerName* MUST be set to NULL.

- *Rid* MUST be set to the RID value retrieved from the Client NTP Request message.
- *Message* MUST refer to the first 48 bytes of the Server NTP Response message.
- *MessageSize* MUST be set to 48.

The **NetrlogonComputeServerDigest** method computes two crypto-checksums using the pair of passwords associated with the trusted account given by the RID.

If the Netlogon Remote Protocol method fails, the server SHOULD<22> fail the authentication and ignore the request without responding.

The server retrieves a 1-bit key selector from the most significant bit of the **Key Identifier** subfield of the **Authenticator** field in the Client NTP Request message. If the 1-bit key selector is set to 0, the server selects the crypto-checksum that was computed using the new password. If the 1-bit key selector is set to 1, the server selects the crypto-checksum that was computed using the old password.<23><24>

The server sets the **Authenticator** field of the Server NTP Response message as described in section 2.2.2, writing the computed crypto-checksum into the **Crypto-Checksum** subfield of the **Authenticator**.

3.2.5.1.2 Non-Domain Controller Processing of the Authenticated Server NTP Response

If the server is not a DC or an **RODC**, the server MUST ignore the client request without responding.

3.2.5.2 LocalClockDispersion Processing

If the ADM element **sys.refid** is set to "LOCL", then the server MUST set the Root Dispersion (**sys.rootdispersion**) ADM element to the value of the **LocalClockDispersion** element. The **sys.refid** and **sys.rootdispersion** ADM elements are defined in [RFC1305] section 3.2.1, "Common Variables".

3.2.6 Timer Events

No timer events are used.

3.2.7 Other Local Events

No additional events are used.

4 Protocol Examples

The following example shows a successful authenticated time synchronization sequence using the NTP Authentication Extensions between a client and a server.

Note that the following packet diagrams illustrate the entire NTP packet, not just the **Authenticator** field in the NTP Authentication Extensions. The NTP data format is defined in the base NTP protocol, as specified in [RFC1305] Appendix A. The **Authenticator** field related to the NTP Authentication Extensions is specified in section 2.2. The packet diagrams use fictitious values for the **Key Identifier** and **Crypto-Checksum** fields.

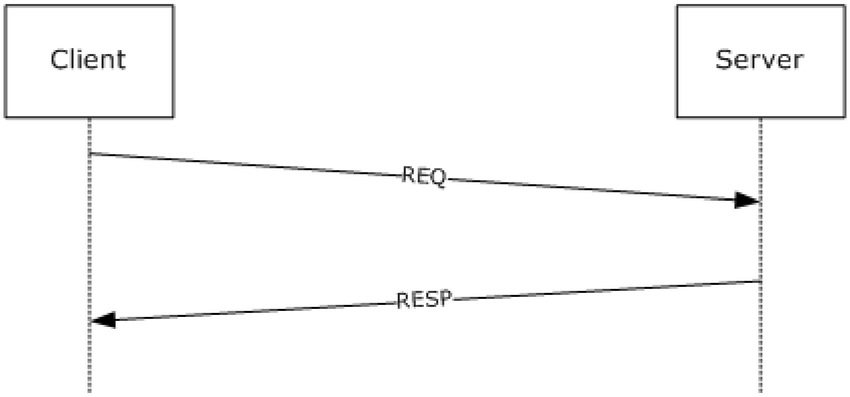

Figure 7: Message sequence

1. The client waits for the client polling timer to expire.

2. The client composes a Client NTP Request message. It sets the **Key Identifier** subfield of the extension part with the RID and the **Key Selector** value, and sets the **Crypto-Checksum** subfield to 0. The client sends the Client NTP Request message to the server.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="2">LI</td><td colspan="3">VN</td><td colspan="3">Mode=3</td><td colspan="8">Stratum</td><td colspan="8">Poll Interval</td><td colspan="8">Precision</td></tr>
<tr><td colspan="32">Root Delay</td></tr>
<tr><td colspan="32">Root Dispersion</td></tr>
<tr><td colspan="32">Reference Clock Identifier</td></tr>
<tr><td colspan="32">Reference Timestamp</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Original Timestamp</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Receive Timestamp
...
Transmit Timestamp
...
0x00 0x35 0x7B 0x9D
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

3. After receiving the request, the server verifies that the received message size is 68 bytes. If it is not, the server either drops the request (if the message size does not equal 48 bytes) or treats it as an unauthenticated request (if the message size is 48 bytes). Assuming that the received message size is 68 bytes, the server extracts the RID from the received message. The server uses it to call the **NetrLogonComputeServerDigest** method (as specified in [MS-NRPC] section 3.5.5.8.2) to compute the crypto-checksums and select the crypto-checksum based on the most significant bit of the **Key Identifier** subfield from the received message, as specified in section 3.2.5. The server then sends a response to the client, setting the **Key Identifier** field to 0 and the **Crypto-Checksum** field to the computed crypto-checksum.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
LI		VN			Mode=4			Stratum								Poll Interval								Precision							
Root Delay																															
Root Dispersion																															
Reference Clock Identifier																															
Reference Timestamp																															
...																															
Original Timestamp																															
...																															
Receive Timestamp																															

...
Transmit Timestamp
...
0x00 0x00 0x00 0x00
0x5E 0xBD 0xA9 0x0E 0xB2 0x35 0x54 0x9A 0xB2 0xA8 0xB3 0x4F 0x50 0x1D 0x62 0x20

4. The client receives the response and, after it verifies that the message size is 68 bytes, computes the crypto-checksums, as specified in section 3.1.5. The client finds that one of the calculated crypto-checksums matches the crypto-checksum in the response. The response is, therefore, authenticated successfully, and the client synchronizes its time.

5 Security

5.1 Security Considerations for Implementers

The cryptographic hash is calculated using the Netlogon Remote Protocol message protection methods. The methods use the MD5 algorithm, which is considered vulnerable to brute-force collision attacks. For more information on MD5 collisions, see [MD5Collision].

NTP Authentication Extensions provide secure Server NTP Response messages but do not prevent invalid ones. For example, when the NTP server's own time is misconfigured, a Server NTP Response message is sent with an invalid time to the client. Such an invalid Server NTP Response message could lead to the client being synchronized to an invalid time. The client SHOULD check the time difference between the client and the server, and SHOULD synchronize time only if the time difference is within some reasonable range.<25>

5.2 Index of Security Parameters

Security parameter	**Section**
Keys and key identifier	3.1.3, 3.1.5, and 3.2.5
Client NTP Request message	2.2.1
Server NTP Response message	2.2.2

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 released service packs:

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

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

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

<1> Section 1.3.2: Windows 2000 extends time synchronization based on SNTP, as specified in [RFC2030]. Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 extend time synchronization based on NTP [RFC1305].

<2> Section 2.2: Windows implements the NTP.MINPOLL and NTP.MAXPOLL elements in the Windows registry by using the following registry values (respectively).

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	MinPollInterval
Type	REG_DWORD

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	MaxPollInterval
Type	REG_DWORD

These elements can be set by using the Remote Registry Protocol [MS-RRP].

<3> Section 2.2.1: Windows clients set this field to 0, and Windows servers ignore this field.

<4> Section 2.2.2: In Windows Server 2003, Windows domain controllers set this field to 0. In Windows 2000, Windows Server 2008, and Windows Server 2008 R2, Windows domain controllers set this field to the value of the **Key Identifier** subfield of the Client NTP Request message.

<5> Section 2.2.2: A server running Windows 2000 Server will return a response that contains a **Reference Timestamp** value equal to the value sent by the client. Additionally, a server running Windows 2000 Server will return a response that contains a **Root Dispersion** value equal to the value sent by the client when the server is unsynchronized.

<6> Section 3.1.1: On Windows 2000 the NtpServer registry value was named "LocalNTP". It exists in the same location with the same value.

<7> Section 3.1.2.1: In Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the minimum polling interval and the maximum polling interval vary between domain roles (member machine versus domain controller). By default, for a member machine acting as an NTP client, the minimum polling interval is 10 and the maximum polling interval is 15; for a domain controller acting as an NTP client, the minimum polling interval is 6 and the maximum polling interval is 10. These interval values are expressed (as in [RFC1305] section 3.2.7) in units of seconds and are exponents to a power of two; thus, the default minimum polling interval for a domain controller is 2 ^ 6 = 64 seconds, and the default maximum polling interval is 2 ^ 10 = 1,024 seconds.

[RFC1305] section 3.2.7 defines constants that specify the minimum (NTP.MINPOLL) and maximum (NTP.MAXPOLL) values permissible for a client's polling interval. The Windows implementation defines different constants for the minimum and maximum permissible values. These constants are used to validate any values specified in configuration for the minimum polling interval and maximum polling interval. The following table shows the definitions of maximum (NTP.MAXPOLL) and minimum (NTP.MINPOLL) permissible values for a client's maximum and minimum polling intervals for different Windows versions.

Windows version	NTP.MAXPOLL: Domain controllers	NTP.MAXPOLL: Member /Standalone machines	NTP.MINPOLL: Domain controllers	NTP.MINPOLL: Member/Standalone machines
Windows XP	15	15	6	10
Windows Server 2003	10	15	6	10
Windows Vista	10	15	6	10
Windows Server 2008	10	15	6	10
Windows 7	10	15	6	10
Windows Server 2008 R2	10	15	6	10

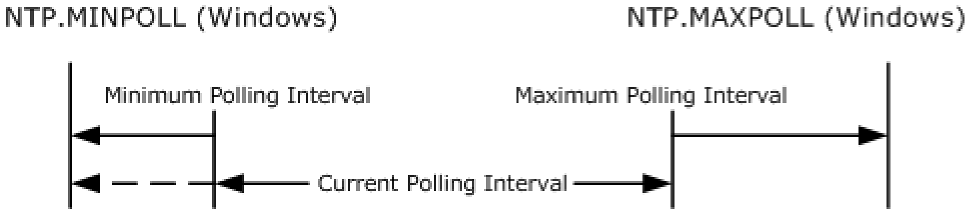

Figure 8: Polling intervals

In Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the **Poll Interval** (as specified in [RFC1305] Appendix A) is initialized to NTP.MINPOLL. If the client continuously receives valid responses, the **Poll Interval** is incremented from NTP.MINPOLL to no more than NTP.MAXPOLL. If the client fails to receive a valid response after three consecutive attempts, the **Poll Interval** is decremented. If the client continues to fail to receive valid responses, the **Poll Interval** is decremented further below the minimum polling interval but never falls below the value defined for NTP.MINPOLL by Windows.

After eight consecutive failures to receive a valid response, the client pauses its synchronization attempts for a "back-off" interval (15 minutes), after which it returns to its initial **Poll Interval**. The back-off interval is doubled for each subsequent occurrence of eight consecutive failures. This doubling occurs no more than six times for a maximum back-off interval of no more than 960 minutes.

In Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the client always incorrectly sets the **Poll Interval** field of the first Client NTP Request message to the value defined for NTP.MAXPOLL by Windows.

Windows 2000 SNTP clients do not implement a true minimum or maximum polling interval. Instead, Windows 2000 clients initially poll by default every 45 minutes (the **Poll Interval** value in the SNTP message is set to 11 for this phase). After three successful poll operations, Windows 2000 clients jump to polling every 8 hours (the **Poll Interval** value is 14 for this phase). After every unsuccessful poll attempt, the interval reverts to 45 minutes.

<8> Section 3.1.3.1: Windows implementation imposes no constraints on the **LargePhaseOffset**, **HoldPeriod**, **SpikeWatchPeriod**, **SpecialPollInterval**, **ResolvePeerBackoffMinutes**, and **ResolvePeerBackoffMaxTimes** element values.

<9> Section 3.1.3.1: Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 do not use "VMTP" for the **sys.refid** element. In Windows 7 and Windows Server 2008 R2, the VMTP value is used when the client or server is in a Windows virtual environment. The determination of whether the client or server is in a Windows virtual environment is a local-only process that is specific to the Microsoft implementation of its virtual environment.

<10> Section 3.1.5.1: Windows 2000 clients do not use the most significant bit of the **Key Identifier** subfield and always set the most significant bit to 0. In Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the client sets the **Key Identifier** subfield as described in section 2.2.1. The most significant bit is initialized to the value of the **Key Selector** abstract element.

The client sets the **Crypto-Checksum** subfield as described in section 2.2.1.

<11> Section 3.1.5.1: In Windows 2000, the client always sets the **Mode** field of its Client NTP Request messages to 0x3 ("Client").

<12> Section 3.1.5.1: In Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the **NetrLogonComputeClientDigest** method, as specified in [MS-NRPC] section 3.5.5.8.3, generates only two crypto-checksums for the current and previous passwords.

<13> Section 3.1.5.2: Windows 2000 servers return the **Reference Timestamp** value from the client request in the response.

<14> Section 3.1.5.2: Windows 2000 clients do not set the **Reference Timestamp** value to 0xAAAAAAAA and do not process Test 6.

<15> Section 3.2.1: Windows 2000 implements only the following values. Note that in Windows 2000, the string names of the values were "Reliable_Time_Source_No" and "Reliable_Time_Source_Yes" with identical semantics.

Value	Meaning
Time_Source_No 0x00	Never advertise as a reliable time source.
Time_Source_Yes 0x01	Always advertise as a reliable time source.

<16> Section 3.2.1: Windows 2000 exposes this ADM element via the following registry key

Attribute	Value
Key Location	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config
Name	ReliableTimeSource
Type	REG_DWORD

This element can be set by using the Remote Registry Protocol [MS-RRP].

<17> Section 3.2.1: The **ResponseMode** element is valid only on Windows 2000.

<18> Section 3.2.3: Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 do not use "VMTP" for the **sys.refid** element. In Windows 7 and Windows Server 2008 R2, the VMTP value is used when the client or server is in a Windows virtual environment. The determination of whether the client or server is in a Windows virtual environment is a local-only process that is specific to the Microsoft implementation of its virtual environment.

<19> Section 3.2.3.1: Windows 2000 performs the following initialization:

- If the machine is a DC or the **ResponseMode** abstract data model element is set to Response_Mode_Yes, then the implementation sets the "Time service is running" bit to 1.
- If the **AnnounceFlags** abstract data model element is set to Time_Source_Yes and either the machine is a DC or the **ResponseMode** abstract data model element is set to Response_Mode_Yes, then the implementation sets the "Time service with clock hardware is running" bit to 1.

<20> Section 3.2.5.1: Windows 2000 Server does not process the Client NTP Request message when the NTP message length is 48 bytes.

<21> Section 3.2.5.1.1: According to [RFC1305], **Receive Timestamp**, **Originate Timestamp**, and **Poll Interval** must be updated for every received NTP message. However, the Windows implementation of the NTP protocol ignores packets with invalid data or invalid headers. An NTP message is marked as having invalid data if it fails any of tests 1 through 4 documented in [RFC1305] section 3.4.4. An NTP message is marked as having an invalid header if it fails any of tests 5 through 8 documented in [RFC1305] section 3.4.4.

<22> Section 3.2.5.1.1: Windows NTP servers in Windows 2000, Windows XP, and Windows Server 2003 do not honor the above "SHOULD". Instead, they respond to the request. In Windows 2000, the server responds with a Server NTP Response message without an **Authenticator** field if authentication fails. In Windows XP and Windows Server 2003, the server responds with a Server NTP Response message that includes an **Authenticator** field in which the **Crypto-Checksum** subfield is set to zero. In either case, the client will consider the Server NTP Response message to be an authentication failure.

In the Windows Server 2008 and Windows Server 2008 R2, in the case of the read-only domain controller (RODC) as the server, if the RODC does not store the cryptographic key locally, the server validates the RID. If the RID identifies a valid object, the server forwards the original Client NTP Request message to its own time source, which must be a writable domain controller. The writable domain controller that has the cryptographic key authenticates the client's request instead. On receiving the response from the writable domain controller, the RODC forwards the response to the client. This process is known as "chaining". If the RID is not identified as a valid object, the server fails the authentication and ignores the request without responding.

<23> Section 3.2.5.1.1: In the situation where the machine account has only a current password (that is, an old password does not yet exist) and a client requests a digest computed using the old password, Windows computes the digest using the current password. Windows 2000 is a special case in that it returns an unauthenticated response when an old password does not exist.

<24> Section 3.2.5.1.1: On Windows servers, machine accounts do not keep a password history and therefore have only a current password. Only domain trust accounts keep the password history; therefore, a domain trust account can have an old password and a current password. In the absence of an old password, the current password is used (for both the 0 and the 1 values of the 1-bit key selector).

<25> Section 5.1: The client accepts any Server NTP Response message regardless of the time difference in authenticated NTP time synchronization inside a Windows domain.

7 Change Tracking

This section identifies changes that were made to the [MS-SNTP] protocol document between the January 2011 and February 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

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 or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

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 **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.1.1 Abstract Data Model	57722 Added information for the NtpServer abstract data model element.	Y	Content updated.

8 Index