

Internet Engineering Task Force  
Internet-Draft  
Updates: 5905 (if approved)  
Intended status: Standards Track  
Expires: 21 April 2022

M. Lichvar  
Red Hat  
A. Malhotra  
Boston University  
18 October 2021

NTP Interleaved Modes  
draft-ietf-ntp-interleaved-modes-07

Abstract

This document extends the specification of Network Time Protocol (NTP) version 4 in RFC 5905 with special modes called the NTP interleaved modes, that enable NTP servers to provide their clients and peers with more accurate transmit timestamps that are available only after transmitting NTP packets. More specifically, this document describes three modes: interleaved client/server, interleaved symmetric, and interleaved broadcast.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Requirements Language . . . . .	4
2. Interleaved Client/server mode . . . . .	4
3. Interleaved Symmetric mode . . . . .	9
4. Interleaved Broadcast mode . . . . .	10
5. Protocol Failures . . . . .	11
6. Security Considerations . . . . .	13
7. IANA Considerations . . . . .	14
8. Acknowledgements . . . . .	14
9. References . . . . .	14
9.1. Normative References . . . . .	14
9.2. Informative References . . . . .	15
Authors' Addresses . . . . .	15

## 1. Introduction

RFC 5905 [RFC5905] describes the operations of NTPv4 in a client/server, symmetric, and broadcast mode. The transmit and receive timestamps are two of the four timestamps included in every NTPv4 packet used for time synchronization.

For a highly accurate and stable synchronization, the transmit and receive timestamp should be captured close to the beginning of the actual transmission and the end of the reception respectively. An asymmetry in the timestamping causes the offset measured by NTP to have an error.

There are at least four options where a timestamp of an NTP packet may be captured with a software NTP implementation running on a general-purpose operating system:

1. User space (software)
2. Network device driver or kernel (software)
3. Data link layer (hardware - MAC chip)

#### 4. Physical layer (hardware - PHY chip)

Software timestamps captured in user space in the NTP implementation itself are least accurate. They do not include system calls used for sending and receiving packets, processing and queuing delays in the system, network device drivers, and hardware. Hardware timestamps captured at the physical layer are most accurate.

A transmit timestamp captured in the driver or hardware is more accurate than the user-space timestamp, but it is available to the NTP implementation only after it sent the packet using a system call. The timestamp cannot be included in the packet itself unless the driver or hardware supports NTP and can modify the packet before or during the actual transmission.

The protocol described in RFC 5905 does not specify any mechanism for a server to provide its clients and peers with a more accurate transmit timestamp that is known only after the transmission. A packet that strictly follows RFC 5905, i.e. it contains a transmit timestamp corresponding to the packet itself, is said to be in basic mode.

Different mechanisms could be used to exchange timestamps known after the transmission. The server could respond to each request with two packets. The second packet would contain the transmit timestamp corresponding to the first packet. However, such a protocol would enable a traffic amplification attack, or it would use packets with an asymmetric length, which would cause an asymmetry in the network delay and an error in the measured offset.

This document describes an interleaved client/server, interleaved symmetric, and interleaved broadcast mode. In these modes, the server sends a packet which contains a transmit timestamp corresponding to the transmission of the previous packet that was sent to the client or peer. This transmit timestamp can be captured in any software or hardware component involved in the transmission of the packet. Both servers and clients/peers are required to keep some state specific to the interleaved mode.

An NTPv4 implementation that supports the client/server and broadcast interleaved modes interoperates with NTPv4 implementations without this capability. A peer using the symmetric interleaved mode does not fully interoperate with a peer which does not support it. The mode needs to be configured specifically for each symmetric association.

The interleaved modes do not change the NTP packet header format and do not use new extension fields. The negotiation is implicit. The protocol is extended with new values that can be assigned to the origin and transmit timestamp. Servers and peers check the origin timestamp to detect requests conforming to the interleaved mode. A response can be valid only in one mode. If a client or peer that does not support interleaved mode received a response conforming to the interleaved mode, it would be rejected as bogus.

An explicit negotiation would require a new extension field. RFC 5905 does not specify how servers should handle requests with an unknown extension field. The original use of extension fields was authentication with Autokey [RFC5906], which cannot be negotiated. Some existing implementations do not respond to requests with unknown extension fields. This behavior would prevent clients from reliably detecting support for the interleaved mode.

Requests and responses cannot always be formed in interleaved mode. It cannot be used exclusively. Servers, clients, and peers that support the interleaved mode need to support also the basic mode.

This document assumes familiarity with RFC 5905.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Interleaved Client/server mode

The interleaved client/server mode is similar to the basic client/server mode. The difference between the two modes is in the values saved to the origin and transmit timestamp fields.

The origin timestamp is a cookie which is used to detect that a received packet is a response to the last packet sent in the other direction of the association. It is a copy of one of the timestamps from the packet to which it is responding, or zero if it is not a response. Servers following RFC 5905 ignore the origin timestamp in client requests. A server response which does not have a matching origin timestamp is called bogus.

A client request in the basic mode has an origin timestamp equal to the transmit timestamp from the last valid server response, or is zero (which indicates the first request of the association). A

server response in the basic mode has an origin timestamp equal to the transmit timestamp from the client request. The transmit timestamp in the response corresponds to the transmission of the response in which the timestamp is contained.

A client request in the interleaved mode has an origin timestamp equal to the receive timestamp from the last valid server response. A server response in the interleaved mode has an origin timestamp equal to the receive timestamp from the client request. The transmit timestamp in the response corresponds to the transmission of the previous response which had the receive timestamp equal to the origin timestamp from the request.

A server which supports the interleaved mode needs to save pairs of local receive and transmit timestamps. The server SHOULD discard old timestamps to limit the amount of memory needed to support clients using the interleaved mode. The server MAY separate the timestamps by IP addresses, but it SHOULD NOT separate them by port numbers to support clients that change their port between requests, as recommended in RFC 9109 [RFC9109].

The server MAY restrict the interleaved mode to specific IP addresses and/or authenticated clients.

Both servers and clients that support the interleaved mode MUST NOT send a packet that has a transmit timestamp equal to the receive timestamp in order to reliably detect whether received packets conform to the interleaved mode. One way to ensure that is to increment the transmit timestamp by 1 unit (i.e. about 1/4 of a nanosecond) if the two timestamps are equal, or a new timestamp can be generated.

The transmit and receive timestamps in server responses need to be unique to prevent two different clients from sending requests with the same origin timestamp and the server responding in the interleaved mode with an incorrect transmit timestamp. If the timestamps are not guaranteed to be monotonically increasing, the server SHOULD check that the transmit and receive timestamps are not already saved as a receive timestamp of a previous request (from the same IP address if the server separates timestamps by addresses), and generate a new timestamp if necessary.

When the server receives a request from a client, it SHOULD respond in the interleaved mode if the following conditions are met:

1. The request does not have a receive timestamp equal to the transmit timestamp.

2. The origin timestamp from the request matches the local receive timestamp of a previous request that the server has saved (for the IP address if it separates timestamps by addresses).

A response in the interleaved mode MUST contain the transmit timestamp of the response which contained the receive timestamp matching the origin timestamp from the request. The server SHOULD drop the timestamps after sending the response. The receive timestamp MUST NOT be used again to detect a request conforming to the interleaved mode.

If the conditions are not met (i.e. the request is not detected to conform to the interleaved mode), the server MUST NOT respond in the interleaved mode. The server MAY always respond in the basic mode. In any case, the server SHOULD save the new receive and transmit timestamps.

The first request from a client is always in the basic mode and so is the server response. It has a zero origin timestamp and zero receive timestamp. Only when the client receives a valid response from the server, it will be able to send a request in the interleaved mode.

The protocol recovers from packet loss. When a client request or server response is lost, the client will use the same origin timestamp in the next request. The server can respond in the interleaved mode if it still has the timestamps corresponding to the origin timestamp. If the server already responded to the timestamp in the interleaved mode, or it had to drop the timestamps for other reasons, it will respond in the basic mode and save new timestamps, which will enable an interleaved response to the subsequent request. The client SHOULD limit the number of requests in the interleaved mode between server responses to prevent processing of very old timestamps in case a large number of consecutive requests is lost.

An example of packets in a client/server exchange using the interleaved mode is shown in Figure 1. The packets in the basic and interleaved mode are indicated with B and I respectively. The timestamps  $t1^{\sim}$ ,  $t3^{\sim}$  and  $t11^{\sim}$  point to the same transmissions as  $t1$ ,  $t3$  and  $t11$ , but they may be less accurate. The first exchange is in the basic mode followed by a second exchange in the interleaved mode. For the third exchange, the client request is in the interleaved mode, but the server response is in the basic mode, because the server did not have the pair of timestamps  $t6$  and  $t7$  (e.g. they were dropped to save timestamps for other clients using the interleaved mode).

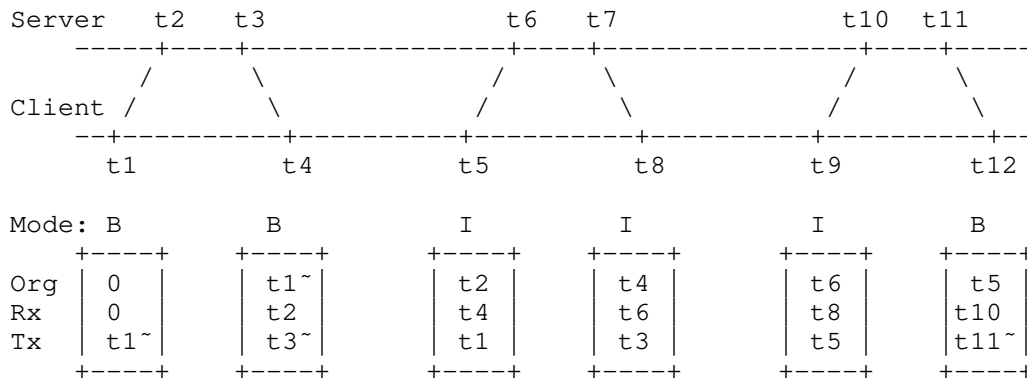


Figure 1: Packet timestamps in interleaved client/server mode

When the client receives a response from the server, it performs the tests described in RFC 5905. Two of the tests are modified for the interleaved mode:

1. The check for duplicate packets SHOULD compare both receive and transmit timestamps in order to not drop a valid response in the interleaved mode if it follows a response in the basic mode and they contain the same transmit timestamp.
2. The check for bogus packets SHOULD compare the origin timestamp with both transmit and receive timestamps from the request. If the origin timestamp is equal to the transmit timestamp, the response is in the basic mode. If the origin timestamp is equal to the receive timestamp, the response is in the interleaved mode.

The client SHOULD NOT update its NTP state when an invalid response is received, to not lose the timestamps which will be needed to complete a measurement when the subsequent response in the interleaved mode is received.

If the packet passed the tests and conforms to the interleaved mode, the client can compute the offset and delay using the formulas from RFC 5905 and one of two different sets of timestamps. The first set is RECOMMENDED for clients that filter measurements based on the delay. The corresponding timestamps from Figure 1 are written in parentheses.

T1 - local transmit timestamp of the previous request (t1)

T2 - remote receive timestamp from the previous response (t2)

T3 - remote transmit timestamp from the latest response (t3)

T4 - local receive timestamp of the previous response (t4)

The second set gives a more accurate measurement of the current offset, but the delay is much more sensitive to a frequency error between the server and client due to a much longer interval between T1 and T4.

T1 - local transmit timestamp of the latest request (t5)

T2 - remote receive timestamp from the latest response (t6)

T3 - remote transmit timestamp from the latest response (t3)

T4 - local receive timestamp of the previous response (t4)

Clients MAY filter measurements based on the mode. The maximum number of dropped measurements in the basic mode SHOULD be limited in case the server does not support or is not able to respond in the interleaved mode. Clients that filter measurements based on the delay will implicitly prefer measurements in the interleaved mode over the basic mode, because they have a shorter delay due to a more accurate transmit timestamp (T3).

The server MAY limit saving of the receive and transmit timestamps to requests which have an origin timestamp specific to the interleaved mode in order to not waste resources on clients using the basic mode. Such an optimization will delay the first interleaved response of the server to a client by one exchange.

A check for a non-zero origin timestamp works with SNTP clients that always set the timestamp to zero and clients that implement NTP data minimization [I-D.ietf-ntp-data-minimization]. From the server's point of view, such clients start a new association with each request.

To avoid searching the saved receive timestamps for non-zero origin timestamps from requests conforming to the basic mode, the server can encode in low-order bits of the receive and transmit timestamps below precision of the clock a flag indicating whether the timestamp is a receive timestamp. If the server receives a request with a non-zero origin timestamp which does not indicate it is a receive timestamp of the server, the request does not conform to the interleaved mode and it is not necessary to perform the search and/or save the new receive and transmit timestamp.



### 3. Interleaved Symmetric mode

The interleaved symmetric mode uses the same principles as the interleaved client/server mode. A packet in the interleaved symmetric mode has a transmit timestamp which corresponds to the transmission of the previous packet sent to the peer and an origin timestamp equal to the receive timestamp from the last packet received from the peer.

To enable synchronization in both directions of a symmetric association, both peers need to support the interleaved mode. For this reason, it SHOULD be disabled by default and enabled with an option in the configuration of the active side of the association.

In order to prevent the peer from matching the transmit timestamp with an incorrect packet when the peers' transmissions do not alternate (e.g. they use different polling intervals) and a previous packet was lost, the use of the interleaved mode in symmetric associations requires additional restrictions.

Peers which have an association need to count valid packets received between their transmissions to determine in which mode a packet should be formed. A valid packet in this context is a packet which passed all NTP tests for duplicate, replayed, bogus, and unauthenticated packets. Other received packets may update the NTP state to allow the (re)initialization of the association, but they do not change the selection of the mode.

A peer A SHOULD send a peer B a packet in the interleaved mode only when all of the following conditions are met:

1. The peer A has an active association with the peer B which was specified with the option enabling the interleaved mode, OR the peer A received at least one valid packet in the interleaved mode from the peer B.
2. The peer A did not send a packet to the peer B since it received the last valid packet from the peer B.
3. The previous packet that the peer A sent to the peer B was the only response to a packet received from the peer B.

The first condition is needed for compatibility with implementations that do not support or are not configured for the interleaved mode. The other conditions prevent a missing response from causing a mismatch between the remote transmit (T2) and local receive timestamp (T3), which would cause a large error in the measured offset and delay.

An example of packets exchanged in a symmetric association is shown in Figure 2. The minimum polling interval of the peer A is twice as long as the maximum polling interval of the peer B. The first packets sent by the peers are in the basic mode. The second and third packet sent by the peer A is in the interleaved mode. The second packet sent by the peer B is in the interleaved mode, but the following packets sent by the peer B are in the basic mode, because multiple responses are sent per request.

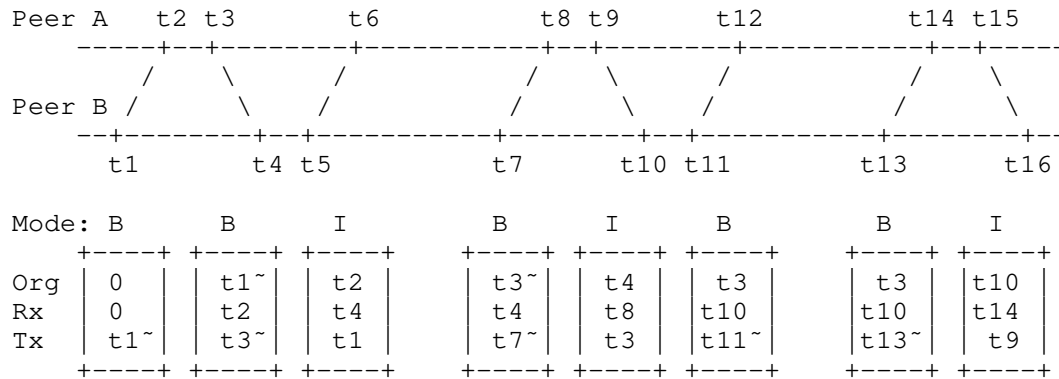


Figure 2: Packet timestamps in interleaved symmetric mode

If the peer A has no association with the peer B and it responds with symmetric passive packets, it does not need to count the packets in order to meet the restrictions, because each request has at most one response. The peer SHOULD process the requests in the same way as a server which supports the interleaved client/server mode. It MUST NOT respond in the interleaved mode if the request was not in the interleaved mode.

The peers SHOULD compute the offset and delay using one of the two sets of timestamps specified in the client/server section. They MAY switch between them to minimize the interval between T1 and T4 in order to reduce the error in the measured delay.

#### 4. Interleaved Broadcast mode

A packet in the interleaved broadcast mode contains two transmit timestamps. One corresponds to the packet itself and is saved in the transmit timestamp field. The other corresponds to the previous packet and is saved in the origin timestamp field. The packet is compatible with the basic mode, which uses a zero origin timestamp.

An example of packets sent in the broadcast mode is shown in Figure 3.

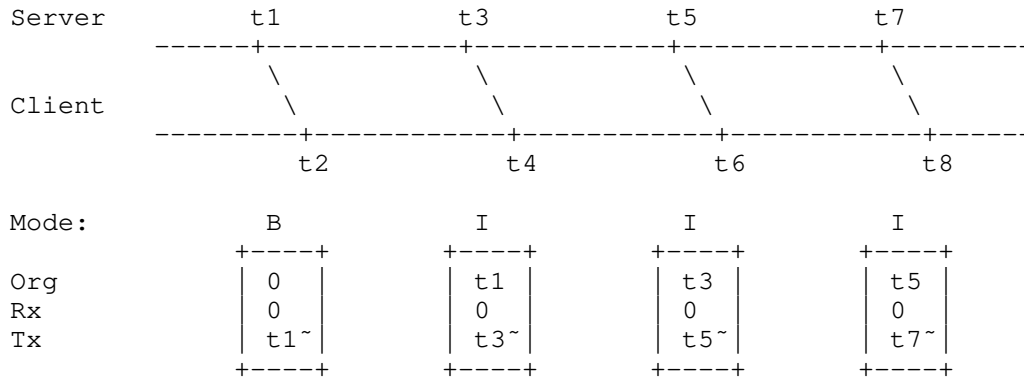


Figure 3: Packet timestamps in interleaved broadcast mode

A client which does not support the interleaved mode ignores the origin timestamp and processes all packets as if they were in the basic mode.

A client which supports the interleaved mode SHOULD check if the origin timestamp is not zero to detect packets in the interleaved mode. The client SHOULD also compare the origin timestamp with the transmit timestamp from the previous packet to detect lost packets. If the difference is larger than a specified maximum (e.g. 1 second), the packet SHOULD NOT be used for synchronization in the interleaved mode.

The client SHOULD compute the offset using the origin timestamp from the received packet and the local receive timestamp of the previous packet. If the client needs to measure the network delay, it SHOULD use the interleaved client/server mode.

### 5. Protocol Failures

An incorrect client implementation of the basic mode (RFC 5905) can work reliably with servers that implement only the basic mode, but the protocol can fail intermittently with servers that implement the interleaved mode.

If the client sets the origin timestamp to other values than the transmit timestamp from the last valid server response, or zero, the origin timestamp can match a receive timestamp of a previous server response (possibly to a different client), causing an unexpected interleaved response. The client is expected to drop the response as bogus. If it did not check for bogus packets, it would be vulnerable to off-path attacks.

If the client set the origin timestamp to a constant non-zero value, this mismatch would be expected to happen once per the NTP era (about 136 years) if the NTP server was responding at the maximum rate needed to go through all timestamp values (about 2 billion responses per second). With lower rates of requests the chance of hitting a server timestamp decreases proportionally.

The worst case of this failure would be a client that specifically sets the origin timestamp to the server's receive timestamp, i.e. the client accidentally implemented the interleaved mode, but it does not accept interleaved responses. This client would still be able to synchronize its clock. It would drop interleaved responses as bogus and set the origin timestamp to the receive timestamp from the last valid response in the basic mode. As servers are required to not respond twice to the same origin timestamp in the interleaved mode, at least every other response would be in the basic mode and accepted by the client.

Intermittent protocol failures can be caused also by an incorrect server implementation of the interleaved mode. A server which does not ensure the receive and transmit timestamps in its responses are unique in a sufficiently long interval can misinterpret requests formed correctly in the basic mode as interleaved and respond in the interleaved mode. The response would be dropped by the client as bogus.

A duplicated server receive timestamp can cause an expected interleaved response to contain a transmit timestamp which does not correspond to the transmission of the previous response from which the client copied the receive timestamp to the origin timestamp in the request, but a different response which contained the same receive timestamp. The response would be accepted by the client with a small error in the transmit timestamp equal to the difference between the transmit timestamps of the two different responses. As the two requests to which the responses responded were received at the same time (according to the server's clock), the two transmissions would be expected to be close to each other and the difference between them would be comparable to the error a basic response normally has in its transmit timestamp.

One reason for a duplicated server timestamp can be a large backward step of the server's clock. If the timestamps the server has saved do not fully cover the second pass of the clock over the repeated interval, two requests received in different passes of the clock can get the same receive timestamp. The client which made the first request can get the transmit timestamp corresponding to the transmission of the second response. From the server's point of view, the error of the transmit timestamp would be still small, but

from the client's point of view the server already failed when it made the step as it was serving wrong time before or after the step with a much larger error than the error caused by the protocol failure.

## 6. Security Considerations

The security considerations of time protocols in general are discussed in RFC 7384 [RFC7384], and specifically the security considerations of NTP are discussed in RFC 5905.

Security issues that apply to the basic modes apply also to the interleaved modes. They are described in The Security of NTP's Datagram Protocol [SECNTP].

Clients and peers SHOULD NOT leak the receive timestamp in packets sent to other peers or clients (e.g. as a reference timestamp) to prevent off-path attackers from easily getting the origin timestamp needed to make a valid response in the interleaved mode.

Clients using the interleaved mode SHOULD randomize all bits of both receive and transmit timestamps, as recommended for the transmit timestamp in the NTP client data minimization [I-D.ietf-ntp-data-minimization], to make it more difficult for off-path attackers to guess the origin timestamp in the server response.

The client data minimization cannot be fully implemented in the interleaved mode. The origin timestamp cannot be zeroed out, which makes the clients more vulnerable to tracking as they move between networks.

Attackers can force the server to drop its timestamps in order to prevent clients from getting an interleaved response. They can send a large number of requests, send requests with a spoofed source address, or replay an authenticated request if the interleaved mode is enabled only for authenticated clients. Clients SHOULD NOT rely on servers to be able to respond in the interleaved mode.

Protecting symmetric associations in the interleaved mode against replay attacks is even more difficult than in the basic mode. In both modes, the NTP state needs to be protected between the reception of the last non-replayed response and transmission of the next request in order for the request to contain the origin timestamp expected by the peer. The difference is in the timestamps needed to complete a measurement. In the basic mode only one valid response is needed at a time and it is used as soon as it is received, but the interleaved mode needs two consecutive valid responses. The NTP state needs to be protected all the time to not lose the timestamps which are needed to complete the measurement when the second response is received.

## 7. IANA Considerations

This memo includes no request to IANA.

## 8. Acknowledgements

The interleaved modes described in this document are based on the implementation written by David Mills in the NTP project (<http://www.ntp.org>). The specification of the broadcast mode is based purely on this implementation. The specification of the symmetric mode has some modifications. The client/server mode is specified as a new mode compatible with the symmetric mode, similarly to the basic symmetric and client/server modes.

The authors would like to thank Theresa Enghardt, Daniel Franke, Benjamin Kaduk, Erik Kline, Tal Mizrahi, Steven Sommars, Harlan Stenn, and Kristof Teichel for their useful comments.

## 9. References

### 9.1. Normative References

- [I-D.ietf-ntp-data-minimization]  
Franke, D. F. and A. Malhotra, "NTP Client Data Minimization", Work in Progress, Internet-Draft, draft-ietf-ntp-data-minimization-04, 25 March 2019, <<https://www.ietf.org/archive/id/draft-ietf-ntp-data-minimization-04.txt>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 9.2. Informative References

- [RFC5906] Haberman, B., Ed. and D. Mills, "Network Time Protocol Version 4: Autokey Specification", RFC 5906, DOI 10.17487/RFC5906, June 2010, <<https://www.rfc-editor.org/info/rfc5906>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.
- [RFC9109] Gont, F., Gont, G., and M. Lichvar, "Network Time Protocol Version 4: Port Randomization", RFC 9109, DOI 10.17487/RFC9109, August 2021, <<https://www.rfc-editor.org/info/rfc9109>>.
- [SECNTP] Malhotra, A., Gundy, M. V., Varia, M., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", 2016, <<http://eprint.iacr.org/2016/1006>>.

## Authors' Addresses

Miroslav Lichvar  
Red Hat  
Purkynova 115  
612 00 Brno  
Czech Republic

Email: [mlichvar@redhat.com](mailto:mlichvar@redhat.com)

Aanchal Malhotra  
Boston University  
111 Cummington St  
Boston, 02215  
United States of America

Email: [aanchal4@bu.edu](mailto:aanchal4@bu.edu)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: 21 December 2024

M. Lichvar  
Red Hat  
19 June 2024

Network Time Protocol Version 5  
draft-ietf-ntp-ntpv5-02

Abstract

This document describes the version 5 of the Network Time Protocol (NTP).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 December 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.



## Table of Contents

1.	Introduction . . . . .	2
1.1.	Requirements Language . . . . .	3
2.	Basic Concepts . . . . .	3
3.	Data Types . . . . .	4
4.	Message Format . . . . .	5
5.	Extension Fields . . . . .	9
5.1.	Draft Identification Extension Field . . . . .	10
5.2.	Padding Extension Field . . . . .	10
5.3.	MAC Extension Field . . . . .	11
5.4.	Reference IDs Request and Response Extension Fields . . . . .	11
5.5.	Server Information Extension Field . . . . .	13
5.6.	Correction Extension Field . . . . .	13
5.7.	Reference Timestamp Extension Field . . . . .	17
5.8.	Monotonic Receive Timestamp Extension Field . . . . .	17
5.9.	Secondary Receive Timestamp Extension Field . . . . .	18
6.	Measurement Modes . . . . .	19
7.	Client Operation . . . . .	22
8.	Server Operation . . . . .	24
9.	Network Time Security with NTPv5 . . . . .	26
10.	NTPv5 Negotiation in previous NTP versions . . . . .	27
11.	Acknowledgements . . . . .	28
12.	IANA Considerations . . . . .	28
13.	Security Considerations . . . . .	30
14.	References . . . . .	30
14.1.	Normative References . . . . .	30
14.2.	Informative References . . . . .	30
	Author's Address . . . . .	31

## 1. Introduction

Network Time Protocol (NTP) is a protocol which enables computers to synchronize their clocks over network. Time is distributed from primary time servers to clients, which can be servers for other clients, and so on. Clients can use multiple servers simultaneously.

NTPv5 is similar to NTPv4 [RFC5905]. The main differences are:

1. The protocol specification (this document) describes only the on-wire protocol. Filtering of measurements, security mechanisms, source selection, clock control, and other algorithms, are out of scope.

2. For security reasons, NTPv5 drops support for the symmetric active, symmetric passive, broadcast, control, and private modes. The symmetric and broadcast modes are vulnerable to replay attacks. The control and private modes can be exploited for denial-of-service traffic amplification attacks. Only the client and server modes remain in NTPv5.
3. Timestamps are clearly separated from values used as cookies.
4. NTPv5 messages can be extended only with extension fields. The MAC field is wrapped in an extension field.
5. Extension fields can be of any length, even indivisible by 4, but are padded to a multiple of 4 octets. Extension fields specified for NTPv4 are compatible with NTPv5.
6. NTPv5 adds support for other timescales than UTC.
7. The NTP era number is exchanged in the protocol, which extends the unambiguous interval of the client from 136 years to about 35000 years.
8. NTPv5 adds interleaved mode to provide clients with more accurate transmit timestamps.
9. NTPv5 works with sets of reference IDs to prevent synchronization loops over multiple hosts.
10. Resolution of the root delay and root dispersion fields is improved from about 15 microseconds to about 4 nanoseconds.
11. Clients do not leak information about their clock (e.g. timestamps).

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 2. Basic Concepts

The distance to the reference time sources in the hierarchy of servers is called stratum. Primary time servers, which are synchronized to the reference clocks, are stratum 1, their clients are stratum 2, and so on.

Root delay measures the total delay on the path to the reference time source used by the primary time server. Each client on the path adds to the root delay the NTP delay measured to the server it considers best for synchronization. The delay includes network delays and any delays between timestamping of NTP messages and their actual reception and transmission. Half of the root delay estimates the maximum error of the clock due to asymmetries in the delay.

Root dispersion estimates the maximum error of the clock due to the instability of the clocks on the path and instability of NTP measurements. Each server on the path adds its own dispersion to the root dispersion. Different clock models can be used. In a simple model, the clock can have a constant dispersion rate, e.g. 15 ppm as used in NTPv4.

The sum of the root dispersion and half of the root delay is called root distance. It is the estimated maximum error of the clock, taking into account asymmetry in delay and stability of clocks and measurements.

Servers have randomly generated reference IDs to enable detection and prevention of synchronization loops.

### 3. Data Types

NTPv5 uses few different data types. They are all in the network order. Beside signed and unsigned integers, it has also the following fixed-point types:

#### time16

A 16-bit signed fixed-point type containing values in seconds. It has 1 signed integer bit (i.e. it is just the sign) and 15 fractional bits. The minimum value is the fraction  $-32767/32768$  (almost -1 second), the maximum value is  $32767/32768$  (almost 1 second), and the resolution is about 30 microseconds. The type has a special value of  $0x8000$ , which indicates an unknown value or value that is too large to be represented by this type.

#### time32

A 32-bit unsigned fixed-point type containing values in seconds. It has 4 bits describing the unsigned integral part and 28 bits describing the fractional part. The maximum value is 16 seconds and the resolution is about 3.7 nanoseconds. Note that this is different from the 32-bit time format in NTPv4.

#### timestamp64

A 64-bit unsigned fixed-point type containing a timestamp describes in seconds. It has 32 signed integer bits and 32

fractional bits. It spans an interval of about 136 years and has a resolution of about 0.23 nanoseconds. It can be used in different timescales. In the UTC timescale it is the number of SI seconds since 1 Jan 1972 plus 2272060800 (number of seconds since 1 Jan 1900 assuming 86400-second days), excluding leap seconds. Timestamps in the TAI timescale are the same except they include leap seconds and extra 10 seconds for the original difference between TAI and UTC in 1972, when leap seconds were introduced. A value of 0 indicates an unknown or invalid timestamp. One interval covered by the type is called an NTP era. The era starting at the epoch is era number 0, the following era is number 1, and so on.

Some fields use a logarithmic scale, where an 8-bit signed integer represents the rounded  $\log_2$  value of seconds. For example, a  $\log_2$  value of 4 is  $2^4$  (2 to the power of 4, 16) seconds, or a  $\log_2$  value of -2 is  $2^{-2}$  (0.25 seconds).

#### 4. Message Format

NTPv5 servers and clients exchange messages as UDP datagrams. Clients send requests to servers and servers send them back responses. The format of the UDP payload is shown in Figure 1.

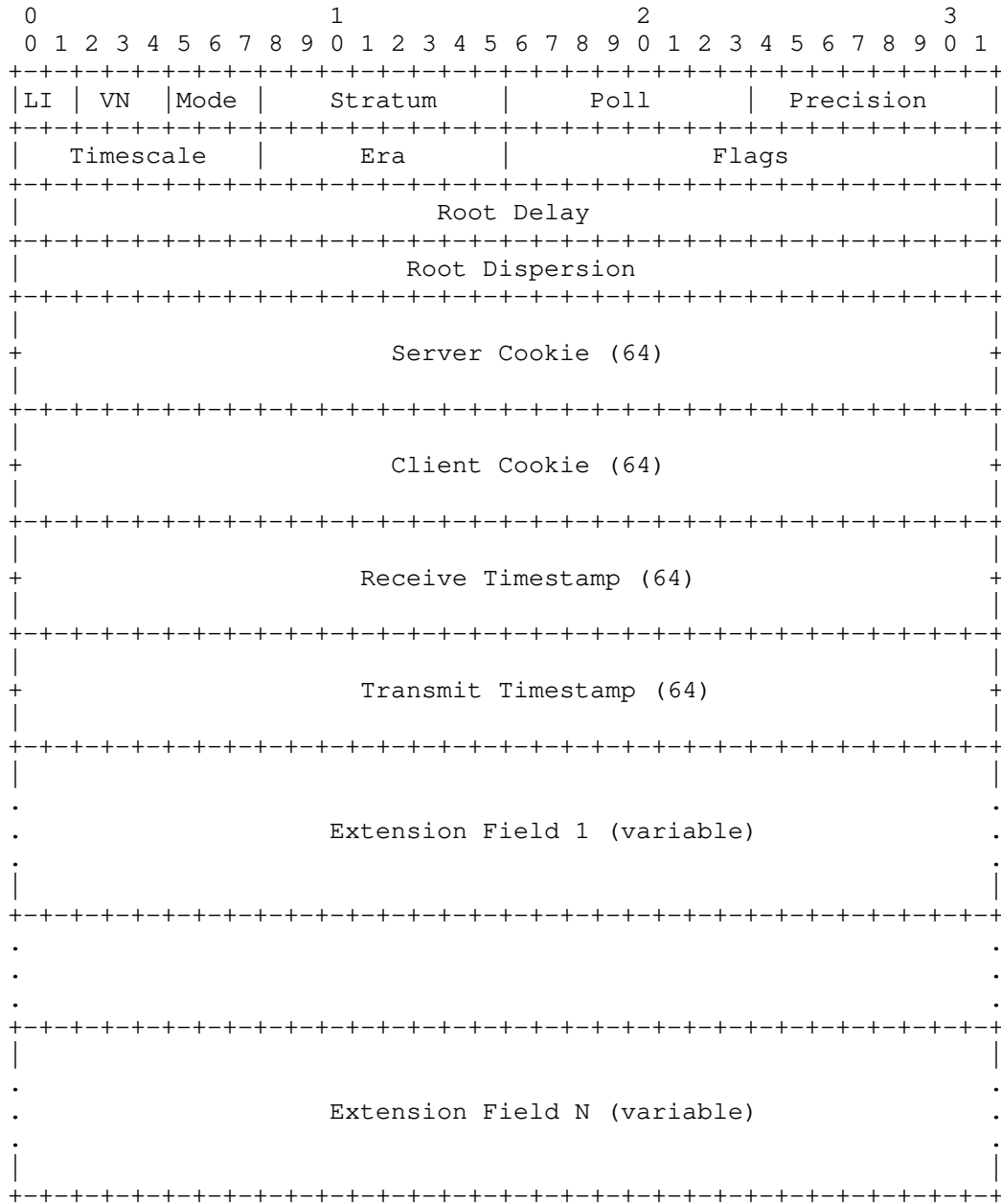


Figure 1: Format of NTPv5 messages

Each NTPv5 message has a header containing the following fields:

**Leap indicator (LI)**

A 2-bit field which can have the following values: 0 (normal), 1 (leap second inserted at the end of the month), 2 (leap second deleted at the end of the month), 3 (not synchronized). The values 1 and 2 are set at most 14 days in advance before the leap second and only if not using a leap-smearred timescale. In requests it is always 0.

**Version Number (VN)**

A 3-bit field containing the value 5.

**Mode**

A 3-bit field containing the value 3 (request) or 4 (response).

**Stratum**

An 8-bit field containing the stratum of the server. Primary time servers have a stratum of 1, their clients have a stratum of 2, and so on. The value of 0 indicates an unknown or infinite stratum. In requests it is always 0. When 0 in a response, it indicates the server was unable or unwilling to provide valid time information. Servers advertising a stratum above 16 should not be synchronized to except when the client is explicitly configured to do so by the end-user.

**Poll**

An 8-bit signed integer containing the polling interval as a rounded log<sub>2</sub> value in seconds. In requests it is the current polling interval. In responses it is the minimum allowed polling interval. This field is valid even when stratum is 0. A value of 127 indicates the server does not want to hear from the client again.

**Precision**

An 8-bit signed integer containing the precision of the timestamps included in the message as a rounded log<sub>2</sub> value in seconds. In requests, which do not contain any timestamps, it is always 0.

**Timescale**

An 8-bit identifier of the timescale. In requests it is the requested timescale. In responses it is the timescale of the receive and transmit timestamps. Defined values are:

0: UTC

1: TAI

2: UT1

### 3: Leap-smearred UTC

#### Era

An 8-bit unsigned NTP era number corresponding to the receive timestamp. In requests it is always 0.

#### Flags

A 16-bit integer that can contain the following flags:

##### 0x1: Unknown leap

In requests it is 0. In responses a value of 1 indicates the server does not have a time source which provides information about leap seconds and the client should interpret the Leap Indicator as having only two possible values: synchronized (0) and not synchronized (3).

##### 0x2: Interleaved mode

In requests a value of 1 is a request for a response in the interleaved mode. In responses a value of 1 indicates the response is in the interleaved mode.

##### 0x4: Authentication NAK

In requests it is 0. In responses a value of 1 indicates that the server failed to authenticate the request. A server setting this bit SHOULD also set the stratum of the message to 0.

#### Root Delay

A field using the time32 type. In responses it is the server's root delay. In requests it is always 0.

#### Root Dispersion

A field using the time32 type. In responses it is the server's root dispersion. In requests it is always 0.

#### Server Cookie

A 64-bit field containing a number generated by the server which enables the interleaved mode. In requests it is 0, or a copy of the server cookie from the last response.

#### Client Cookie

A 64-bit field containing a random number generated by the client. Responses contain a copy of the field from the corresponding request, which allows the client to verify that the responses are related to the requests.

Receive Timestamp

A field using the timestamp64 type. In requests it is always 0. In responses it is the time when the request was received by the server. The timestamp corresponds to the end of the reception.

Transmit Timestamp

A field using the timestamp64 type. In requests it is always 0. In responses it is the server's time denoting the beginning of the transmission of a response to the client. Which response it refers to depends on the selected mode (basic or interleaved). See Measurement Modes (Section 6) for detail.

The header has 48 octets, which is the minimum length of a valid NTPv5 message. A message can contain optional extension fields (zero or more). The maximum length is not specified, but the length MUST be divisible by 4.

5. Extension Fields

The format of NTPv5 extension fields is shown in Figure 2.

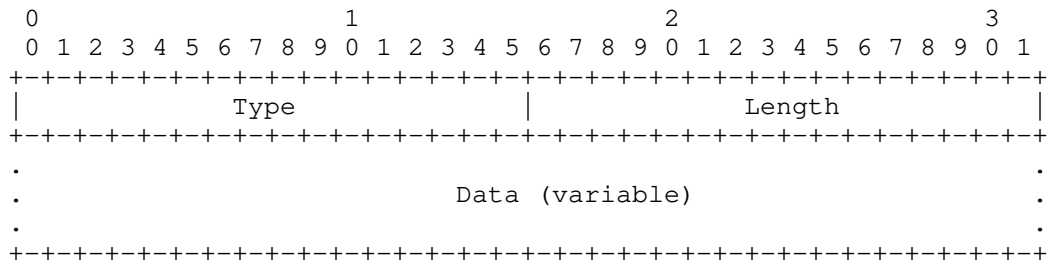


Figure 2: Format of NTPv5 extension fields

Each extension field has a header which contains a 16-bit type and 16-bit length. The length is in octets and it includes the header. The minimum length is 4, i.e. an extension field does not have to contain any data. If the length is not divisible by 4, the extension field is padded with zeros to the smallest multiple of 4 octets.

If a request contains an extension field, the server MUST include this extension field in the response unless the specification of the extension field states otherwise, or the server does not support the extension field. A client can interpret the absence of an expected extension field in a response as an indication that the server does not support the extension field.

Extension fields specified for NTPv4 can be included in NTPv5 messages as specified for NTPv4.



The rest of this section describes extension fields specified for NTPv5. Clients are not required to use or support any of these extension fields, but servers are required to support at least the Padding Extension Field, Server Information Extension field, and if they can be synchronized to other servers, also the Reference IDs Request and Response extension fields to enable detection of synchronization loops.

### 5.1. Draft Identification Extension Field

Note to the editors: this section must be removed before final publication.

This field, with type 0xF5FF, is used to indicate which draft of the specification an implementation is based upon. It MUST be included in NTPv5 requests produced by an implementation based on a draft of this specification, and MUST NOT be included in NTPv5 requests produced by an implementation based on the final version of this specification. Server MUST use this field if and only if responding to a request containing this field and the server is a draft implementation.

The contents of this field MUST be the full name, including version number, of the draft upon which the implementation is based, encoded as an ASCII string. If the server string is longer than the client string, the server MUST truncate it to the length of the client string.

A server SHOULD NOT respond to requests with a draft identification it does not recognize, and SHOULD respond with a response according to the same draft specification as given by the client.

Note: the content of this field MUST NOT be null terminated

### 5.2. Padding Extension Field

This field, with type [[TBD]] (draft: 0xF501), is used by servers to pad the response to the same length as the request if the response does not contain all requested extension fields, or some have a variable length. It can have any length. The data field of the extension field SHOULD contain zeros and it MUST be ignored by the receiver.

This field MUST be supported on servers.

### 5.3. MAC Extension Field

This field, with type `[[TBD]]` (draft: 0xF502), authenticates the NTPv5 message with a symmetric key. Implementations SHOULD use the MAC specified in RFC8573 [RFC8573]. The extension field MUST be the last extension field in the message unless an extension field is specifically allowed to be placed after a MAC or another authenticator field.

### 5.4. Reference IDs Request and Response Extension Fields

Each NTPv5 server has a randomly generated 120-bit reference ID (it will be split into 10 12-bit values). The extension fields described in this section are used to exchange sets of reference IDs in order to detect synchronization loops, i.e. when a client is synchronizing (directly or indirectly) to one of its own clients.

As each client can be synchronized to an unlimited number of servers (and there can be up to 15 strata of servers), the reference IDs are exchanged as a Bloom filter [Bloom] instead of a list to limit the amount of data that needs to be exchanged.

The Bloom filter is an array of 4096 bits. When empty, all bits are zero. To add a reference ID to the filter, the 120-bit value of the reference ID is split into 10 12-bit values and the bits of the array at the 10 positions given by the 12-bit values are set to one.

A server maintains a copy of the filter for each server it is using as an NTP client. The filter provided by the server to clients is the union of the filters (using the bitwise OR operation) of the server's sources selected for synchronization and the server's own reference ID.

If the server uses a previous version of NTP for some of its sources, the reference IDs added to the filter are generated from their IP addresses as the first 120 bits of the MD5 [RFC1321] sum of the address.

A client checking whether the server's set of reference IDs contains the client's own reference ID checks whether the bits at the 10 positions corresponding to the 12-bit values from the reference ID are all set to one.

When a client which serves time to other clients detects a synchronization loop with one of its servers, it SHOULD stop using the server for synchronization. When the client's reference ID is no longer detected in the server's filter, it SHOULD wait for a random number of polling intervals (e.g. between 0 and 4) before selecting the server again. The random delay helps with stabilization of the selection in longer loops.

False positives are possible. The probability of a collision grows with the number of reference IDs in the filter. With 26 reference IDs it is about 1e-12. With 118 IDs it is about 1e-6. The client MAY avoid selecting a server which has too many bits set in the filter (e.g. more than half) to reduce the probability of the collision for its own clients. A client which detected a synchronization loop MAY change its own reference ID to limit the duration of the potential collision.

The filter can be exchanged as a single 512-octet array, or it can be exchanged in smaller chunks over multiple NTP messages, making them shorter, but delaying the detection of the synchronization loop.

The request extension field specifies the offset of the requested chunk in the filter as a number of octets. The requested length of the chunk is given by the length of the extension field. The response extension field MUST have the same length as the request extension field. If the request contains an invalid offset, the extension field MUST be ignored.

The client SHOULD use requests of a constant length for the association to avoid adding a variation to the measured NTP delay.

The format of the Reference IDs Request is shown in Figure 3.

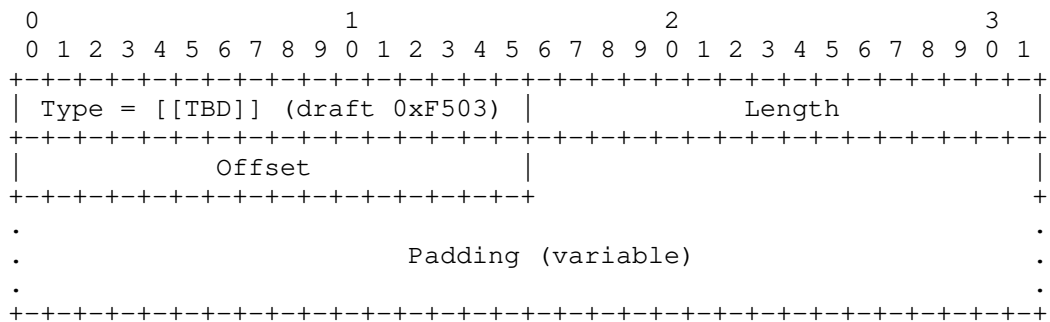


Figure 3: Format of Reference IDs Request Extension Field

The format of the Reference IDs Response is shown in Figure 4.

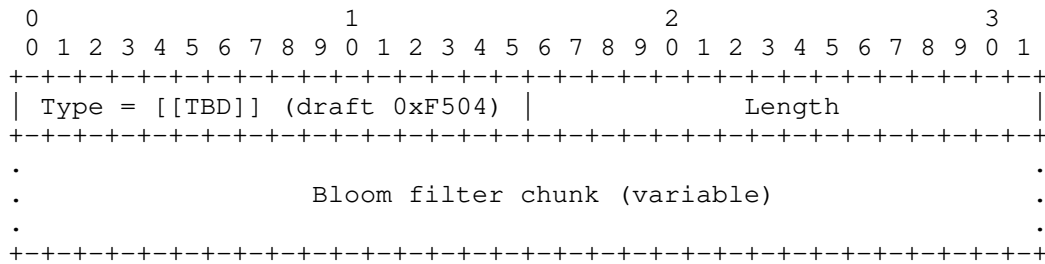


Figure 4: Format of Reference IDs Response Extension Field

These fields MUST be supported on servers which can be synchronized to other NTP servers (i.e. they can be in a synchronization loop).

5.5. Server Information Extension Field

This field provides clients with information about which NTP versions are supported by the server, i.e. whether it can respond to requests conforming to the specific version. It contains a 16-bit field with flags indicating support for NTP versions in the range of 1 to 16, where the least significant bit corresponds to the version 1. The extension field has a fixed length of 8 octets. In requests, all data fields of the extension are 0.

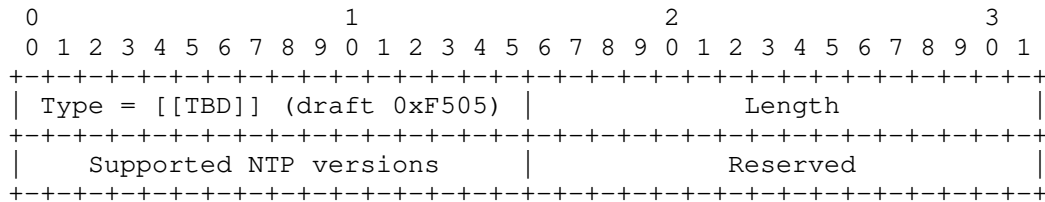


Figure 5: Format of Server Information Extension Field

This field MUST be supported on servers.

5.6. Correction Extension Field

Processing and queueing delays in network switches and routers may be a significant source of jitter and asymmetry in network delay, which has a negative impact on accuracy and stability of clocks synchronized by NTP. A solution to this problem is defined in the Precision Time Protocol (PTP) [IEEE1588], which is a different protocol for synchronization of clocks in networks. In PTP a special type of switch or router, called a Transparent Clock (TC), updates a correction field in PTP messages to account for the time messages spend in the TC. This is accomplished by timestamping the message at

the ingress and egress ports, taking the difference to determine time in the TC and adding this to the Delay Correction. Clients can account for the accumulated Delay Correction to determine a more accurate clock offset and network delay.

The NTPv5 Delay Correction has the same format as the PTP correctionField to make it easier for manufacturers of switches and routers to implement NTP corrections. The format of the Correction Extension Field is shown in Figure 6.

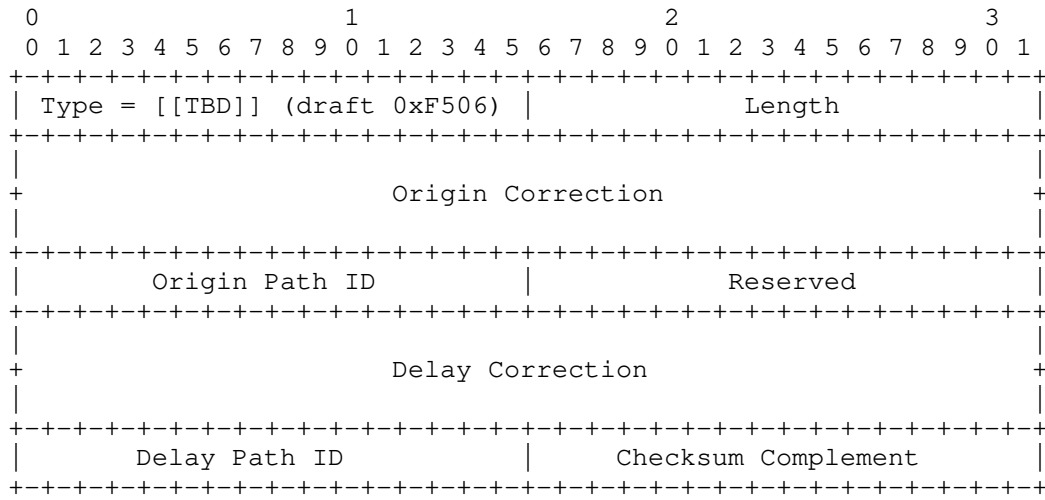


Figure 6: Format of Correction Extension Field

Field Type

The type which identifies the Correction extension field (value TBD).

Length

The length of the extension field, which is 28 octets.

Origin Correction

A field which contains a copy of the accumulated delay correction from the request packet in the NTP exchange.

Origin Path ID

A field which contains a copy of the final path ID from the request packet in the NTP exchange.

Reserved

16 bit reserved for future specification by the IETF. Transmit with all zeros.

#### Delay Correction

A signed fixed-point number of nanoseconds with 48 integer bits and 16 binary fractional bits, which represents the current correction of the network delay that has accumulated for this packet on the path from the source to the destination. The format of this field is identical to the PTP correctionField.

#### Path ID

A 16-bit identification number of the path where the delay correction was updated.

#### Checksum Complement

A field which can be modified in order to keep the UDP checksum of the packet valid. This allows the UDP checksum to be transmitted before the Correction Field is received and modified. The same field is described in RFC 7821 [RFC7821].

An NTP client with enabled support for network corrections SHOULD add the Correction Extension Field to its requests, with all fields of the extension field set to zero. It MUST be the last extension field in the NTP message.

A network device forwarding packets (e.g. a switch or router) with enabled support for NTP corrections MUST modify only packets which meet all of the following requirements:

1. It is an IPv4 or IPv6 packet (e.g. EtherType of 0x0800 or 0x86DD)
2. The IP protocol number is 17 (UDP)
3. The UDP source port or destination port is 123 (NTP)
4. The NTP version is 5
5. The NTP message contains the Correction Extension Field

The network device SHOULD add the difference between the beginning of the NTP message retransmission and the end of the message reception to the Delay Correction value in the Correction Extension Field. Note that this time difference might be negative, e.g. in a cut-through switch. If the packet is transmitted at the same speed as it was received and the length of the packet does not change (e.g. due to adding or removing a VLAN tag), the beginning and end of the interval may correspond to any point of the reception and transmission as long as it is consistent for all forwarded packets of the same length. If the transmission speed or length of the packet is different, the beginning and end of the interval SHOULD correspond to the end of the reception and beginning of the transmission respectively.

The receive timestamp of the ingress port and transmit timestamp of the egress port MUST be from the same clock, or two clocks that are synchronized to each other. The clocks do not need to be synchronized to an external reference if their frequency is accurate enough for the accuracy of measured NTP delay required by the application. The correction field is updated before or during the transmission of the message. It is a one-step end-to-end transparent clock in the PTP terminology.

The network device SHOULD have a randomly generated 16-bit ID number assigned to each of its ports. When it modifies the Correction Extension Field, it SHOULD update the Path ID field of the extension field by adding to it the values of the incoming and outgoing port ID. The Path ID values can be used by clients to determine if the NTP request and response have likely traversed the same network path.

If the network device modified any fields of the Correction Extension Field, it MUST also update the Checksum Complement field in order to keep the existing UDP checksum valid, or update the UDP checksum in the UDP header itself. The network device MUST NOT modify any other data in the UDP payload.

If an NTP server supports the Correction Extension Field and receives a request which contains this extension field, it SHOULD include the extension field in the response. If it is included, it MUST be the last extension field in the message. It MUST copy the Delay Correction and Path ID from the request to the Origin Correction and Origin Path ID fields in the response respectively. It SHOULD set the Delay Correction and Path ID fields of the response to zero.

The Correction Extension Field is required to be the last extension field in the message to provide an indicator at a fixed offset from the ending of the message reception (which might simplify hardware implementation of the correction) and to avoid being covered by the

MAC Extension Field, or other authenticator extension fields (e.g. Network Time Security). It is the exception in the requirement specified for the MAC Extension Field. If the corrections were covered by the authenticator fields, the network devices would need to have access to the keys and have a significant additional complexity in order to update the authenticator fields when they modify the Correction Extension Field.

As the Correction Extension Field is not protected, NTP clients MUST validate the corrections before their application as specified in Measurement Modes (Section 6). Clients MUST ignore an unexpected Correction Extension Field in the response, i.e. if it was not included in the request.

5.7. Reference Timestamp Extension Field

This field contains the time of the last update of the clock. It has a fixed length of 12 octets. In requests, that timestamp is always 0.

(Is this really needed? It was mostly unused in NTPv4.)

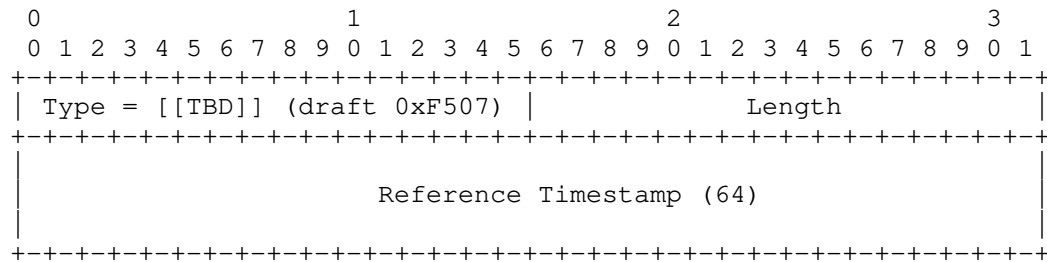


Figure 7: Format of Reference Timestamp Extension Field

5.8. Monotonic Receive Timestamp Extension Field

When a clock is synchronized to a time source, there is a compromise between time (phase) accuracy and frequency accuracy, because the frequency of the clock has to be adjusted to correct time errors that accumulate due to the frequency error (e.g. caused by changes in the temperature of the crystal). Faster corrections of time can minimize the time error, but increase the frequency error, which transfers to clients using that clock as a time source and increases their frequency and time errors. This issue can be avoided by transferring time and frequency separately using different clocks.



The Monotonic Receive Timestamp Extension Field contains an extra receive timestamp with a 32-bit epoch ID captured by a clock which does not have corrected phase and can better transfer frequency than the clock which captures the receive and transmit timestamps in the header. The extension field has a constant length of 16 octets. In requests, the counter and timestamp are always 0.

The epoch ID is a random number which is changed when frequency transfer needs to be restarted, e.g. due to a step of the clock.

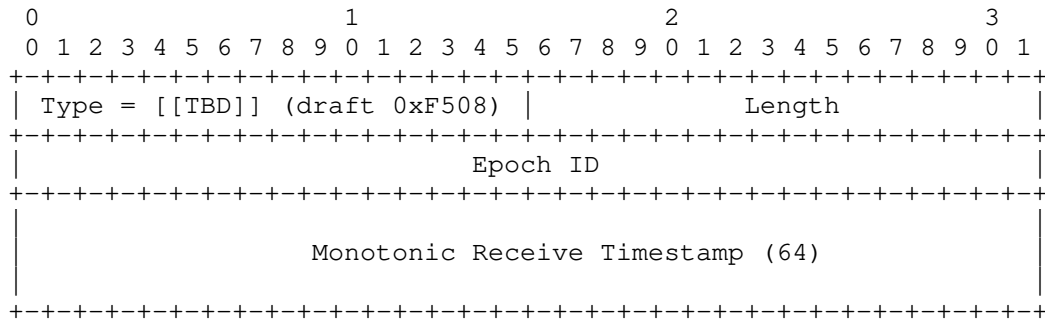


Figure 8: Format of Monotonic Receive Timestamp Extension Field

The client can determine the frequency-transfer offset from the time-transfer offset and difference between the two receive timestamps in the response. It can use the frequency-transfer offset to better control the frequency of its clock, avoiding the frequency error in the server's time-transfer clock.

### 5.9. Secondary Receive Timestamp Extension Field

This extension field provides an additional receive timestamp of the client request in a selected timescale. It enables the client to get the same receive timestamp in different timescales in order to calculate the current offset between the timescales.

In requests, the Timescale field selects the requested timescale. The other data fields in the extension field MUST be set to 0.

The Timescale, Era, and Secondary Receive Timestamp fields in a response have the same meaning as the Timescale, Era, and Receive Timestamp fields in the header respectively.

If the server does not support the requested timescale, it MUST ignore the extension field in the request. If the server supports the timescale, but does not have a reliable timestamp (e.g. due to being close to a leap second), it SHOULD set the timestamp field to 0.

The server MAY provide in this extension field timestamps in timescales which it does not provide in the header, e.g. it can provide UTC in addition to leap-smearred UTC to enable its clients to measure the current smearing offset.

A request MAY contain multiple instances of this extension field, but each timescale MUST be requested at most once, not counting the timescale in the header. The server SHOULD include in its response timestamps in all timescales it supports.

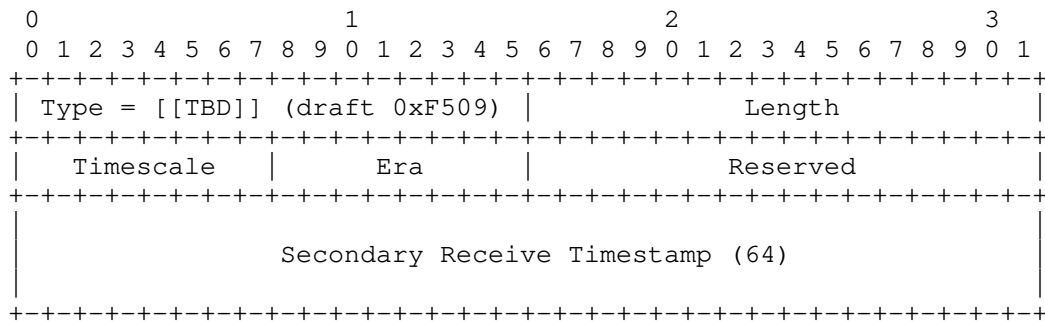


Figure 9: Format of Secondary Receive Timestamp Extension Field

### 6. Measurement Modes

An NTPv5 client needs four timestamps to measure the offset and delay of its clock relative to the server's clock:

1. T1 - client's transmit timestamp of a request
2. T2 - server's receive timestamp of the request
3. T3 - server's transmit timestamp of a response
4. T4 - client's receive timestamp of the response

The offset, delay and dispersion are calculated as:

- \*  $offset = ((T2 + T3) - (T4 + T1)) / 2$
- \*  $delay = |(T4 - T1) - (T3 - T2)|$

\*  $\text{dispersion} = |T4 - T1| * DR$

where

\*  $T1, T2, T3, T4$  are the receive and transmit timestamps of a request and response

\*  $DR$  is the client's dispersion rate

If the Correction Extension Field is used and the corrections are known for both the request and response, a corrected offset and delay is calculated:

\*  $\text{offset}_c = \text{offset} + (Cd - Co) / 2$

\*  $\text{delay}_c = \text{delay} - (Cd + Co - Drx - Dtx) * (1 - FC)$

where

\*  $Co$  is the Origin Correction from the response to the request corresponding to timestamps  $T1$  and  $T2$

\*  $Cd$  is the Delay Correction from the response corresponding to timestamps  $T3$  and  $T4$

\*  $FC$  is the maximum expected frequency error of devices providing the delay corrections (e.g. 100 ppm)

\*  $Drx$  is the time it took to receive the frame containing the response corresponding to  $T3$  and  $T4$

\*  $Dtx$  is the time it took to transmit the frame containing the request corresponding to  $T1$  and  $T2$ . If unknown, it SHOULD be set to  $Drx$ .

The corrected offset and delay MUST NOT be accepted if any of  $\text{delay}_c$ ,  $Co$  and  $Cr$  is negative. The uncorrected delay MUST always be used for calculation of root delay.

The client can make measurements in the basic mode, or interleaved mode if supported on the server. In the basic mode, the transmit timestamp in the server response corresponds to the message which contains the timestamp itself. In the interleaved mode it corresponds to a previous response identified by the server cookie. The interleaved mode enables the server to provide the client with a more accurate transmit timestamp which is available only after the response was formed or sent.

An example of cookies and timestamps in an NTPv5 exchange using the basic mode is shown in Figure 10.

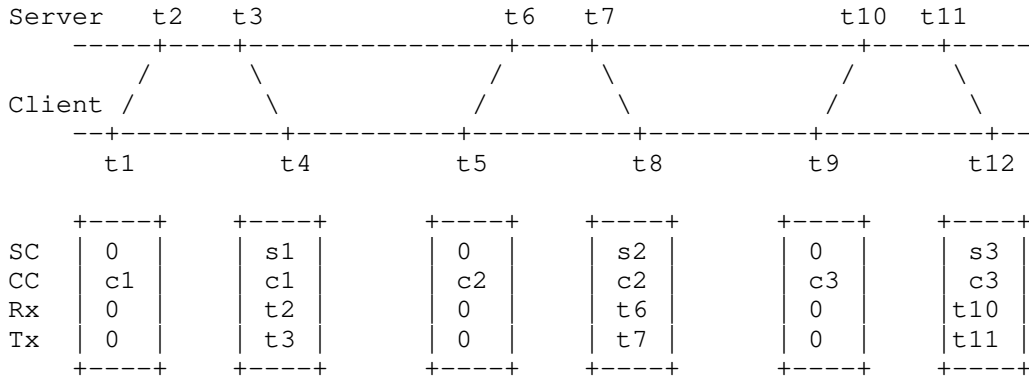


Figure 10: Cookies and timestamps in basic mode

From the three exchanges in this example, the client would use the the following sets of timestamps:

- \* (t1, t2, t3, t4)
- \* (t5, t6, t7, t8)
- \* (t9, t10, t11, t12)

For NTPv4, the interleaved mode is described in NTP Interleaved Modes [I-D.ietf-ntp-interleaved-modes]. The difference between the NTPv5 and NTPv4 interleaved modes is that in NTPv5 it is enabled with a flag and the previous transmit timestamp on the server is identified by the server cookie instead of the receive timestamp.

An example of an NTPv5 exchange using the interleaved mode is shown in Figure 11. The messages in the basic and interleaved mode are indicated with B and I respectively. The timestamps t3' and t11' correspond to the same transmissions as t3 and t11, but they may be less accurate (e.g. due to being captured in software before the transmission). The first exchange is in the basic mode followed by a second exchange in the interleaved mode. For the third exchange, the client request is in the interleaved mode, but the server response is in the basic mode, because the server no longer had the timestamp t7 (e.g. it was dropped to save timestamps for other clients using the interleaved mode).

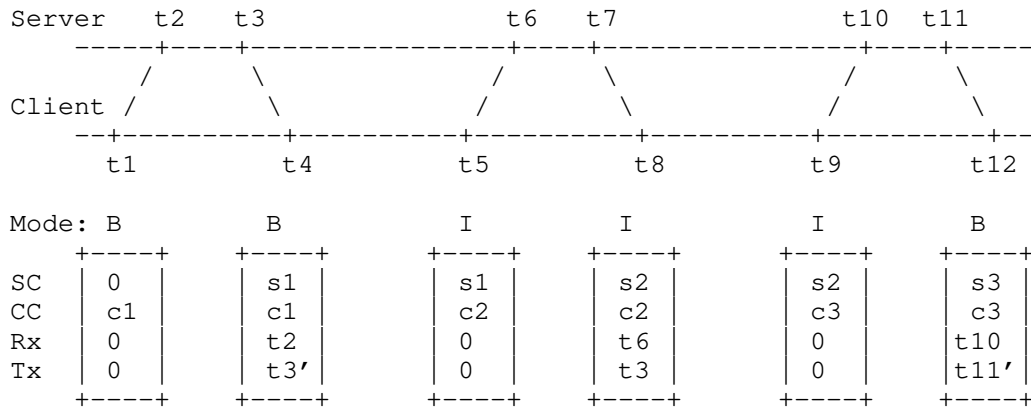


Figure 11: Cookies and timestamps in interleaved mode

From the three exchanges in this example, the client would use the following sets of timestamps:

- \* (t1, t2, t3', t4)
- \* (t1, t2, t3, t4) or (t5, t6, t3, t4)
- \* (t9, t10, t11', t12)

### 7. Client Operation

An NTPv5 client can use one or multiple servers. It has a separate association with each server. It makes periodic measurements of its offset and delay to the server. It can filter the measurements and compare measurements from different servers to select and combine the best servers for synchronization. It can adjust its clock in order to minimize its offset and keep the clock synchronized. These algorithms are not specified in this document.

The polling interval can be adjusted for the network conditions and stability of the clock. When polling a public server on Internet, the client SHOULD use a polling interval of at least 64 seconds, increasing in normal conditions up to at least 1024 seconds to avoid excessive load on the server in case the implementation is used on a very large number of systems.

Each successful measurement provides the client with an offset, delay and dispersion. When combined with the server's root delay and dispersion, it gives the client an estimate of the maximum error.

On each poll, the client:

1. Generates a new random cookie.
2. Formats a request with necessary extension fields and the fields in the header all zero except:
  - \* Version is set to 5.
  - \* Mode is set to 3.
  - \* Scale is set to the timescale in which the client wants to operate.
  - \* Poll is set to the rounded log2 value of the current client's polling interval in seconds.
  - \* Flags are set according to the requested mode. The interleaved mode flag requests the server to save the transmit timestamp of the response and provide the transmit timestamp of a previous response corresponding to the server cookie (if not zero).
  - \* Server cookie is set only in the interleaved mode. It is set to the server cookie from the last valid response, or zero if no such response was received yet or the transmit timestamp of that response would no longer be useful to the client (e.g. after missing too many responses).
  - \* Client cookie is set to the newly generated cookie.
3. Sends the request to the server to the UDP port 123 and captures a transmit timestamp for the packet.
4. Waits for a valid response from the server and captures a receive timestamp. A valid response has version 5, mode 4, client cookie equal to the cookie from the request, and passes authentication if enabled. The client MUST ignore all invalid responses and accept at most one valid response.
5. Checks whether the response is usable for synchronization of the clock. Such a response has a leap indicator not equal to 3, stratum between 0 and 16, root delay and dispersion both smaller than a specific value, e.g. 16 seconds, and timescale equal to the requested timescale. If the response is in a different timescale, the client can switch to the provided timescale, convert the timestamps if the offset between the timescales is known from the Secondary Receive Timestamp Extension Field or other sources, or ignore the response.

6. Saves the server's receive and transmit timestamps. If the client internally counts seconds using a type wider than 32 bits, it SHOULD expand the timestamps with the provided NTP era.
7. Calculates the offset, delay, and dispersion as specified in Measurement Modes (Section 6).

A client which operates as a server for other clients MUST include the Reference IDs Request Extension Field in its requests in order to track reference IDs of its sources. If the server's set of reference IDs contains the client's own reference ID, it SHOULD not select the server for synchronization to avoid a synchronization loop. If the client is requesting the reference IDs in multiple chunks, it SHOULD NOT select the server for synchronization until it received the whole set.

A client which uses multiple servers MUST be able to handle servers providing timestamps in different timescales. It can ignore servers not using the most common or preferred timescale, or convert them to a common timescale if it knows the offsets between them.

If the client synchronizes its clock to a leap-smearred timescale, it MUST NOT apply leap seconds and it SHOULD provide the same timescale to its own clients if it is a server.

The client SHOULD periodically (e.g. every two weeks) refresh IP addresses of all servers specified by hostname to limit the amount of traffic that migrated or decommissioned servers will receive from long-running clients.

## 8. Server Operation

A server receives requests on the UDP port 123. The server MUST support measurements in the basic mode. It MAY support the interleaved mode.

For the basic mode the server does not need to keep any client-specific state. For the interleaved mode it needs to save transmit timestamps and be able to identify them by a cookie.

The server maintains its leap indicator, stratum, root delay, and root dispersion:

- \* Leap indicator MUST be 3 if the clock is not synchronized or its maximum error cannot be estimated with the root delay and dispersion. Otherwise, it MUST be 0, 1, 2, depending on whether a leap second is pending in the next 14 days and, if it is, whether it will be inserted or deleted.

- \* Stratum SHOULD be one larger than stratum of the best server it uses for its own synchronization.
- \* Root delay SHOULD be the best server's root delay in addition to the measured delay to the server.
- \* Root dispersion SHOULD be the best server's root dispersion in addition to an estimate of the maximum drift of its own clock since the last update of the clock.

The server has a randomly generated 120-bit reference ID. It MUST track reference IDs of its servers in order to be able to respond with a Reference IDs Response Extension Field.

For each received request, the server:

1. Captures a receive timestamp.
2. Checks the version in the request. If it is not equal to 5, it MUST either drop the request, or handle it according to the specification corresponding to the protocol version.
3. Drops the request if the format is not valid, mode is not 3, or authentication fails with the MAC Extension Field or another authenticator which does not have a specified response for failed authentication. The server MUST ignore unknown extension fields.
4. Server forms a response with requested extension fields and sets the fields in the header as follows:
  - \* Leap Indicator, Stratum, Root delay, and Root dispersion, are set to the current server's values.
  - \* Version is set to 5.
  - \* Scale is set to the client's requested timescale if it is supported by the server. If not, the server SHOULD respond in any timescale it supports.
  - \* The flags are set as follows:

Unknown leap is set if the server does not know if a leap second is pending in the next 14 days, i.e. it has no source providing information about leap seconds.

Interleaved mode is set if the interleaved mode is



implemented, was requested, and a response in the interleaved mode is possible (i.e. a transmit timestamp is associated with the server cookie).

- \* Era is set to the NTP era of the receive timestamp.
  - \* Server Cookie is set when the interleaved mode is requested and it is supported by the server, even if the response cannot be in the requested mode due to the request having an unknown or zero server cookie. The cookie identifies a more accurate transmit timestamp of the response, which can be retrieved by the client later with another request. The cookie generation is implementation-specific.
  - \* Client Cookie is set to the Client Cookie from the request.
  - \* Receive Timestamp is set to the server's receive timestamp of the request.
  - \* Transmit Timestamp is set to a value which depends on the measurement mode. In the basic mode it is the server's current time when the message is formed. In the interleaved mode it is the transmit timestamp of the previous response identified by the server cookie in the request, captured at some point after the message was formed.
5. Adds the Padding Extension field if necessary to make the length of the response equal to the length of the request.
  6. Drops the response if it is longer than the request to prevent traffic amplification.
  7. Sends the response.
  8. Saves the transmit timestamp and server cookie, if the interleaved mode was requested and is supported by the server.
9. Network Time Security with NTPv5

The Network Time Security [RFC8915] mechanism uses the NTS-KE protocol to establish keys and negotiate the next protocol. NTPv5 can be indicated as the next protocol with identifier [[TBD]] (draft use 0x8001). This can be used by clients and servers to negotiate NTPv5 for an NTS session.

No new NTS-KE records are specified for NTPv5. The records that were specified for NTPv4 (i.e. NTPv4 New Cookie, NTPv4 Server Negotiation, and NTPv4 Port Negotiation) are reused for NTPv5.

The NTS extension fields specified for NTPv4 are compatible with NTPv5. No new extension fields are specified.

(Note to editor: remove this paragraph before publishing.) Client implementations of a draft of this specification MUST provide the identity of the draft implemented as data in an NTS-KE record of type 0x4001, which does not have the critical bit set. The draft identity MUST be encoded as ascii and MUST not contain any trailing 0 bytes. Servers that implement a draft MUST not accept NTPv5 as an option unless they support the specific draft version identified.

#### 10. NTPv5 Negotiation in previous NTP versions

NTPv5 messages are not compatible with NTPv4 and other previous versions of NTP, even if they do not contain any extension fields. Some widely used server NTPv4 implementations are known to accept requests indicating a higher version, interpreting them as NTPv4, and copying the version number from the request to its NTPv4 response. Responses to NTPv5 requests have a zero client cookie, which causes them to fail the NTPv5 validation and be ignored by the client.

The implementations are also known to not respond to requests with an unknown extension field, which prevents an NTPv4 extension field to be specified for NTPv5 negotiation. Instead, the negotiation reuses the reference timestamp field.

An NTP server which supports NTPv5 and also NTPv4, NTPv3, NTPv2, or NTPv1, SHOULD check the reference timestamp in received client-mode requests of the previous NTP versions. If the reference timestamp contains the value 0x4E5450354E545035 ("NTP5NTP5" in ASCII), it SHOULD respond with the same reference timestamp to indicate it supports NTPv5.

Note to the editor: Remove this paragraph before publication. Implementations of a draft of this specification SHOULD use 0x4E54503544524654 ("NTP5DRFT" in ASCII) instead of 0x4E5450354E545035.

When NTPv5 is not expected to be widely supported on servers yet, an NTP client which supports both NTPv5 and a previous NTP version, is not configured to use a specific NTP version, and does not use NTS or other mechanism including negotiation of the NTP version, SHOULD start with the previous version and set the reference timestamp to 0x4e5450354e545035. If the server responds with the same reference timestamp, the client SHOULD switch to NTPv5. If no valid response is received for a number of requests (e.g. 2), the client SHOULD switch back to the original NTP version and stick with it for a larger number of requests (e.g. 256) before trying NTPv5 again.

The special value of the reference timestamp corresponds to 1941-08-24T01:13:25Z in NTP era 0 and 2077-09-29T07:41:41Z in NTP era 1. The negotiation will probably not be needed at that time anymore. If NTPv5 servers and NTPv4-or-older-only clients are still in use and they send a request with the special value by chance, they will get an acceptable response with no side effects. If an NTPv5 client gets the special value by chance from an NTPv4-or-older-only server, it will switch to NTPv5 and back after missing a valid response.

## 11. Acknowledgements

Some ideas were taken from a different NTPv5 design proposed by Daniel Franke.

The author would like to thank Doug Arnold and David Venhoek for their contributions and Dan Drown, Watson Ladd, Hal Murray, Kurt Roeckx, and Ulrich Windl for their suggestions and comments.

## 12. IANA Considerations

IANA is requested to create a new registry for NTPv5 Extension Field Types with initial entries including all entries from the NTPv4 Extension Field Types Registry [RFC5905] and the following NTPv5-specific entries:

Field Type	Meaning	Reference
[[TBD]], selected by IANA from the IETF Review range	Padding	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	MAC	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Reference IDs Request	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Reference IDs Response	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Server Information	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Correction	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Reference Timestamp	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Monotonic Receive Timestamp	[[this memo]]
[[TBD]], selected by IANA from the IETF Review range	Secondary Receive Timestamp	[[this memo]]

Table 1

IANA is requested to allocate the following protocol in the Network Time Security Next Protocols Registry [RFC8915]:

Protocol ID	Protocol Name	Reference
[[TBD]], selected by IANA from the IETF Review range	Network Time Protocol version 5 (NTPv5)	[[this memo]]

Table 2

### 13. Security Considerations

The corrections provided by network devices in the Correction Extension Field are not authenticated. Man-in-the-middle attackers can modify the corrections, but only corrections smaller than the measured delay are accepted by clients. The security impact is comparable to the impact of delaying unmodified NTP messages, which generally cannot be prevented.

### 14. References

#### 14.1. Normative References

- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<https://www.rfc-editor.org/info/rfc1321>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8573] Malhotra, A. and S. Goldberg, "Message Authentication Code for the Network Time Protocol", RFC 8573, DOI 10.17487/RFC8573, June 2019, <<https://www.rfc-editor.org/info/rfc8573>>.

#### 14.2. Informative References

- [Bloom] Bloom, B. H., "Space/Time Trade-Offs in Hash Coding with Allowable Errors", June 1970, <<https://doi.org/10.1145/362686.362692>>.
- [I-D.ietf-ntp-interleaved-modes] Lichvar, M. and A. Malhotra, "NTP Interleaved Modes", Work in Progress, Internet-Draft, draft-ietf-ntp-interleaved-modes-07, 18 October 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-ntp-interleaved-modes-07>>.
- [IEEE1588] Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2019, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems."", November 2019, <<https://www.ieee.org>>.

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC7821] Mizrahi, T., "UDP Checksum Complement in the Network Time Protocol (NTP)", RFC 7821, DOI 10.17487/RFC7821, March 2016, <<https://www.rfc-editor.org/info/rfc7821>>.
- [RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/info/rfc8915>>.

## Author's Address

Miroslav Lichvar  
Red Hat  
Email: [mlichvar@redhat.com](mailto:mlichvar@redhat.com)

Network Time Protocol  
Internet-Draft  
Intended status: Informational  
Expires: 28 July 2024

J. Guessing  
Nederlandse Publieke Omroep  
25 January 2024

NTPv5 Use Cases and Requirements  
draft-ietf-ntp-ntpv5-requirements-04

Abstract

This document describes the use cases, requirements, and considerations that should be factored in the design of a successor protocol to supersede version 4 of the NTP protocol presently referred to as NTP version 5 ("NTPv5"). It aims to define what capabilities and requirements such a protocol possesses, informing the design of the protocol in addition to capturing any working group consensus made in development.

Note to Readers

\_RFC Editor: please remove this section before publication\_

Source code and issues for this draft can be found at <https://github.com/fiestajetsam/draft-guessing-ntp-ntpv5-requirements> (<https://github.com/fiestajetsam/draft-guessing-ntp-ntpv5-requirements>).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 July 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- 1. Introduction . . . . . 3
  - 1.1. Notational Conventions . . . . . 3
- 2. Use Cases and Existing Deployments of NTP . . . . . 3
- 3. Threat Analysis and Modeling . . . . . 4
  - 3.1. Denial of Service and Amplification . . . . . 4
  - 3.2. Accuracy Degradation . . . . . 4
  - 3.3. False Time . . . . . 5
- 4. Requirements . . . . . 5
  - 4.1. Resource Management . . . . . 5
  - 4.2. Data Minimisation . . . . . 6
  - 4.3. Algorithms . . . . . 6
  - 4.4. Timescales . . . . . 7
  - 4.5. Leap seconds . . . . . 7
  - 4.6. Backwards Compatibility with NTS and NTPv4 . . . . . 7
    - 4.6.1. Dependent Specifications . . . . . 8
  - 4.7. Extensibility . . . . . 8
  - 4.8. Security . . . . . 8
- 5. Non-requirements . . . . . 9
  - 5.1. Server Malfeasance Detection . . . . . 9
  - 5.2. Additional Time Information and Metadata . . . . . 9
  - 5.3. Remote Monitoring Support . . . . . 9
- 6. IANA Considerations . . . . . 9
- 7. Security Considerations . . . . . 9
- 8. References . . . . . 9
  - 8.1. Normative References . . . . . 9
  - 8.2. Informative References . . . . . 10
- Appendix A. Acknowledgements . . . . . 11
- Author's Address . . . . . 11



## 1. Introduction

NTP version 4 [RFC5905] has seen active use for over a decade, and within this time period the protocol has not only been extended to support new requirements but has also fallen victim to vulnerabilities that have been used for distributed denial of service (DDoS) amplification attacks. In order to advance the protocol and address these known issues alongside add capabilities for future usage this document defines the current known and applicable use cases in existing NTPv4 deployments and defines requirements for the future.

### 1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Use of time specific terminology used in this document may further be specified in [RFC7384] or NTP specific terminology and concepts within [RFC5905].

## 2. Use Cases and Existing Deployments of NTP

As a protocol, NTP is used synchronise large amounts of computers via both private networks and the open internet, and there are several common scenarios for existing NTPv4 deployments: publicly accessible NTP services such as the NTP Pool [ntp.pool] are used to offer clock synchronisation for end users and embedded devices, ISP-provided servers are used to synchronise devices such as customer-premises equipment. Depending on the network and path these deployments may be affected by variable latency as well as throttling or blocking by providers.

Data centres and cloud computing providers have also deployed and offer NTP services both for internal use and for customers, particularly where the network is unable to offer or does not require the capabilities other protocols can provide, and where there may already be familiarity with NTP. As these deployments are less likely to be constrained by network latency or power, the potential for higher levels of accuracy and precision within the bounds of the protocol are possible, particularly through the use of modifications such as the use of bespoke algorithms.

### 3. Threat Analysis and Modeling

A considerable motivation towards a new version of the protocol is the inclusion of security primitives such as authentication and encryption to bring the protocol in-line with current best practices for protocol design.

There are numerous potential threats to a deployment or network handling traffic time synchronisation protocols that [RFC7384] section 3 describes, which can be summarised into three basic groups: Denial of Service (DoS), degradation of accuracy, and false time, all of which in various forms apply to NTP. However, not all threats apply specifically to NTP directly, most notable attacks on time sources (section 3.2.10) and L2/L3 DoS Attacks (section 3.2.7) as both are outside the scope of the protocol, and the protocol itself cannot provide much in the way of mitigations.

#### 3.1. Denial of Service and Amplification

NTPv4 has previously suffered from DDoS amplification attacks using a combination of IP address spoofing and private mode commands used in some NTP implementations, leading to an attacker being able to direct very large volumes of traffic to a victim IP address. Current mitigations are disabling private mode commands susceptible to attacks and encouraging network operators to implement BCP 38 [RFC2827] as well as source address validation where possible.

The NTPv5 protocol specification should be designed with current best practices for UDP based protocols in mind [RFC8085]. It should reduce the potential amplification factors in request/response payload sizes [drdos-amplification] through the use of padding of payload data, in addition to restricting command and diagnostic modes which could be exploited.

#### 3.2. Accuracy Degradation

The risk that an on-path attacker can systemically delay packets between a client and server exists in all time protocols operating on insecure networks and its mitigations within the protocol are limited for a clock which is not yet synchronised. Increased path diversity and protocol support for synchronisation across multiple heterogeneous sources are likely the most effective mitigations.

### 3.3. False Time

Conversely, on-path attackers who can manipulate timestamps could also speed up a client's clock resulting in drift-related malfunctions and errors such as premature expiration of certificates on affected hosts. An attacker may also manipulate other data in flight to disrupt service and cause de-synchronisation. Additionally attacks via replaying previously transmitted packets can also delay or confuse receiving clocks, impacting ongoing synchronisation.

Message authentication with regular key rotation should mitigate all of these cases; however deployments should consider finding an appropriate compromise between the frequency of rotation to balance the window of attack and the rate of re-keying.

## 4. Requirements

At a high level, NTPv5 should be a protocol that is capable of operating in local networks and over public internet connections where packet loss, delay, and filtering may occur. It should provide both basic time information and synchronisation.

### 4.1. Resource Management

Historically there have been many documented instances of NTP servers receiving ongoing large volumes of unauthorised traffic [ntp-misuse] and the design of NTPv5 must ensure the risk of these can be minimised through the use of signalling unwanted traffic (e.g Kiss of Death) or easily identifiable packet formats which make rate-limiting, filtering, or blocking by firewalls possible.

The protocol's loop avoidance mechanisms SHOULD be able to use identifiers that change over time. Identifiers MUST NOT relate to network topology, in particular such mechanism should not rely on any FQDN, IP address or identifier tied to a public certificate used or owned by the server. Servers SHOULD be able to migrate and change any identifier used as stratum topologies or network configuration changes occur.

An additional identifier mechanism MAY be considered for the purposes of client allow/deny lists, logging and monitoring. Such a mechanism when included, SHOULD be independent of any loop avoidance mechanism, and authenticity requirements SHOULD be considered.

The protocol MUST have the capability for servers to notify clients that the service is unavailable and clients MUST have clearly defined behaviours for honouring this signalling. In addition servers SHOULD be able to communicate to clients that they should reduce their query rate when the server is under high load or has reduced capacity.

Clients SHOULD periodically re-establish connections with servers to prevent maintaining prolonged connectivity to unavailable hosts and give operators the ability to move traffic away from hosts in a timely manner.

The protocol SHOULD have provisions for deployments where Network Address Translation occurs and define behaviours when NAT rebinding occurs. This should also not compromise any DDoS mitigation(s) that the protocol may define.

Client and server protocol modes MUST be supported. Other modes such as symmetric and broadcast MAY be supported by the protocol but SHOULD NOT be required by implementers to implement. Considerations should be made in these modes to avoid implementation vulnerabilities and to protect deployments from attacks.

#### 4.2. Data Minimisation

To minimise ongoing use of deprecated fields and exposing identifying information of implementations and deployments, payload formats SHOULD use the least amount of fields and information where possible, realising that data minimisation and resource management can be at odds with one another. The use of extensions should be preferred when transmitting optional data.

#### 4.3. Algorithms

The use of algorithms describing functions such as clock filtering, selection, and clustering SHOULD have agility, allowing for implementations to develop and deploy new algorithms independently. Signalling of algorithm use or preference SHOULD NOT be transmitted by servers, however essential properties of the algorithm (e.g. precision) SHOULD be obvious.

The working group should consider creating a separate informational document to describe an algorithm to assist with implementation, and consider adopting future documents which describe new algorithms as they are developed. Specifying client algorithms separately from the protocol will allow NTPv5 to meet the needs of applications with a variety of network properties and performance requirements.

#### 4.4. Timescales

The protocol should adopt a linear, monotonic timescale as the basis for communicating time. The format should provide sufficient scale, precision, and resolution to meet or exceed NTPv4's capabilities, and have a roll-over date sufficiently far into the future that the protocol's complete obsolescence is likely to occur first. Ideally it should be similar or identical to the existing epoch and data model that NTPv4 defines to allow for implementations to better support both versions of the protocol, simplifying implementation.

The timescale, in addition to any other time-sensitive information, MUST be sufficient to calculate representations of both UTC and TAI [TF.460-6], noting that UTC itself as the current timescale used in NTPv4 is neither linear nor monotonic unlike TAI. Through extensions the protocol SHOULD support additional timescale representations outside of the main specification, and all transmissions of time data MUST indicate the timescale in use.

#### 4.5. Leap seconds

Transmission of UTC leap second information MUST be included in the protocol in order for clients to generate a UTC representation, but must be transmitted as separate information to the timescale. The specification MUST require that servers transmit upcoming leap seconds greater than 24 hours in linear timescale in advance if that information is known by the server. If the server learns of a leap second less than 24 hours before an upcoming leap second event, it MUST start transmitting the information immediately.

Smearing [google-smear] of leap seconds SHOULD be supported in the protocol, and the protocol MUST support servers transmitting information if they are configured to smear leap seconds and if they are actively doing so. Behaviours for both client and server in handling leap seconds MUST be part of the specification; in particular how clients handle multiple servers where some may use leap seconds and others smearing, that servers should not apply both leap seconds and smearing, as well as details around smearing timescales. Supported smearing algorithms MUST be defined or referenced.

#### 4.6. Backwards Compatibility with NTS and NTPv4

The desire for compatibility with older protocols should not prevent addressing deployment issues or cause ossification of the protocol caused by middleboxes [RFC9065].

Servers that support multiple versions of NTP MUST send a response in the same version as the request as the model of backwards compatibility. This does not preclude servers from acting as a client in one version of NTP and a server in another.

Protocol ossification MUST be addressed to prevent existing NTPv4 deployments which respond incorrectly to clients posing as NTPv5 from causing issues. Forward prevention of ossification (for a potential NTPv6 protocol in the future) should also be taken into consideration.

#### 4.6.1. Dependent Specifications

Many other documents make use of NTP's data formats ([RFC5905] Section 6) for representing time, notably for media and packet timestamp measurements, such as SDP [RFC4566] and STAMP [RFC8762]. Any changes to the data formats should consider the potential implementation complexity that may be incurred.

#### 4.7. Extensibility

The protocol MUST have the capability to be extended; implementations MUST ignore unknown extensions. Unknown extensions received from a lower stratum server SHALL NOT be re-transmitted towards higher stratum servers.

#### 4.8. Security

Data authentication and integrity MUST be supported by the protocol, with optional support for data confidentiality. Downgrade attacks by an in-path attacker must be mitigated. The protocol MUST define at least one common mechanism to ensure interoperability, but should also include support for different mechanisms to support different deployment use cases. Extensions and additional modes SHOULD also incorporate authentication and integrity on data which could be manipulated by an attacker, on-path or off-path.

Upgrading cryptographic algorithms must be supported, allowing for more secure cryptographic primitives to be incorporated as they are developed and as attacks and vulnerabilities with incumbent primitives are discovered.

Intermediate devices such as networking equipment capable of modifying NTP packets, for example to adjust timestamps MUST be able to do so without compromising authentication or confidentiality. Extension fields with separate authentication may be used to facilitate this.

Consideration must be given to how this will be incorporated into any applicable trust model. Downgrading attacks that could lead to an adversary disabling or removing encryption or authentication MUST NOT be possible in the design of the protocol.

## 5. Non-requirements

This section covers topics that are explicitly out of scope.

### 5.1. Server Malfeasance Detection

Detection and reporting of server malfeasance should remain out of scope as [I-D.ietf-ntp-rough-time] already provides this capability as a core functionality of the protocol.

### 5.2. Additional Time Information and Metadata

Previous versions of NTP do not transmit additional time information such as time zone data or historical leap seconds, and NTPv5 should not explicitly add support for it by default as existing protocols (e.g. TZDIST [RFC7808]) already provide mechanisms to do so. This does not prevent however, further extensions enabling this.

### 5.3. Remote Monitoring Support

Largely due to previous DDoS amplification attacks, mode 6 messages which have historically provided the ability for monitoring of servers SHOULD NOT be supported in the core of the protocol. However, it may be provided as a separate extension specification. It is likely that even with a new version of the protocol middleboxes may continue to block this mode in default configurations into the future.

## 6. IANA Considerations

This document makes no requests of IANA.

## 7. Security Considerations

As this document is intended to create discussion and consensus, it introduces no security considerations of its own.

## 8. References

### 8.1. Normative References

- [I-D.ietf-ntp-rougtime]  
Malhotra, A., Langley, A., Ladd, W., and M. Dansarie,  
"Rougtime", Work in Progress, Internet-Draft, draft-ietf-  
ntp-rougtime-08, 18 October 2023,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-ntp-rougtime-08>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering:  
Defeating Denial of Service Attacks which employ IP Source  
Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827,  
May 2000, <<https://www.rfc-editor.org/rfc/rfc2827>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch,  
"Network Time Protocol Version 4: Protocol and Algorithms  
Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010,  
<<https://www.rfc-editor.org/rfc/rfc5905>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in  
Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384,  
October 2014, <<https://www.rfc-editor.org/rfc/rfc7384>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage  
Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,  
March 2017, <<https://www.rfc-editor.org/rfc/rfc8085>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC  
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,  
May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.

## 8.2. Informative References

- [drdos-amplification]  
"Amplification and DRDoS Attack Defense -- A Survey and  
New Perspectives", n.d.,  
<<https://arxiv.org/abs/1505.07892>>.
- [google-smear]  
"Google Leap Smear", n.d.,  
<<https://developers.google.com/time/smea>>.



- [ntp-misuse] "NTP server misuse and abuse", n.d.,  
<[https://en.wikipedia.org/wiki/NTP\\_server\\_misuse\\_and\\_abuse](https://en.wikipedia.org/wiki/NTP_server_misuse_and_abuse)>.
- [ntppool] "pool.ntp.org: the internet cluster of ntp servers", n.d.,  
<<https://www.ntppool.org>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<https://www.rfc-editor.org/rfc/rfc4566>>.
- [RFC7808] Douglass, M. and C. Daboo, "Time Zone Data Distribution Service", RFC 7808, DOI 10.17487/RFC7808, March 2016, <<https://www.rfc-editor.org/rfc/rfc7808>>.
- [RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <<https://www.rfc-editor.org/rfc/rfc8762>>.
- [RFC9065] Fairhurst, G. and C. Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols", RFC 9065, DOI 10.17487/RFC9065, July 2021, <<https://www.rfc-editor.org/rfc/rfc9065>>.
- [TF.460-6] "Standard-frequency and time-signal emissions", n.d., <<https://www.itu.int/rec/R-REC-TF.460-6-200202-I/en>>.

#### Appendix A. Acknowledgements

The author would like to thank Doug Arnold, Hal Murray, Paul Gear, and David Venhoek for contributions to this document, and would like to acknowledge Daniel Franke, Watson Ladd, Miroslav Lichvar for their existing documents and ideas. The author would also like to thank Angelo Moriondo, Franz Karl Achard, and Malcom McLean for providing the author with motivation.

#### Author's Address

James Guessing  
Nederlandse Publieke Omroep  
Netherlands  
Email: [james.ietf@gmail.com](mailto:james.ietf@gmail.com)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: 22 November 2024

M. Lichvar  
Red Hat  
21 May 2024

NTP Over PTP  
draft-ietf-ntp-over-ntp-03

Abstract

This document specifies a transport for the client-server and symmetric modes of the Network Time Protocol (NTP) which encapsulates NTP messages in messages of the Precision Time Protocol (PTP). This transport enables hardware timestamping in network interface controllers which can timestamp only PTP messages and enables delay corrections in PTP transparent clocks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 November 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1.	Introduction . . . . .	2
1.1.	Comparison with PTP . . . . .	3
1.2.	Requirements Language . . . . .	4
2.	PTP transport for NTP . . . . .	5
3.	Network Correction Extension Field . . . . .	6
4.	Acknowledgements . . . . .	9
5.	IANA Considerations . . . . .	9
6.	Implementation Status - RFC EDITOR: REMOVE BEFORE PUBLICATION . . . . .	10
6.1.	chrony . . . . .	10
7.	Security Considerations . . . . .	11
8.	References . . . . .	11
8.1.	Normative References . . . . .	11
8.2.	Informative References . . . . .	12
	Author's Address . . . . .	12

## 1. Introduction

The Precision Time Protocol (PTP) [IEEE1588-2019] was designed for highly accurate synchronization of clocks in local networks. It relies on hardware timestamping supported in all network devices involved in the synchronization (e.g. network interface controllers, switches, and routers) to eliminate the impact of software, processing and queueing delays on accuracy of offset and delay measurements.

PTP was originally designed for multicast communication. Later was added support for unicast messaging, which is useful in larger networks with partial on-path PTP support (e.g. telecom profiles G.8265.1 and G.8275.2).

The Network Time Protocol [RFC5905] does not rely on hardware timestamping support, but implementations can use it if it is available to avoid the impact of software, processing and queueing delays, similarly to PTP. When comparing PTP with the timing modes of NTP, PTP is functionally closest to the NTP broadcast mode.

An issue for NTP is hardware that can specifically timestamp only PTP packets. This limitation comes from a hardware design which can provide receive timestamps only at a limited rate instead of the maximum rate possible at the network link speed. To avoid missing receive timestamps when the interface is receiving other traffic at a high rate, a filter is implemented in the hardware to inspect each received packet and capture a timestamp only for packets that need it.

The hardware filter can be usually configured for specific PTP transport (e.g. UDP over IPv4, UDP over IPv6, 802.3) and sometimes even the PTP message type (e.g. sync message or delay request) to further reduce the timestamping rate on the server or client side in the case of multicast messaging, but it typically cannot be configured to timestamp NTP messages sent to the UDP port 123.

Another issue for NTP is missing hardware support in network switches and routers. With PTP the devices operate either as boundary clocks or transparent clocks. Boundary clocks are analogous to NTP clients that work also as servers for other clients. Transparent clocks are much simpler. They only measure the delay in forwarding of PTP packets and write this delay to the correction field of either the packet itself (one-step mode) or a later packet in the PTP exchange (two-step mode). Transparent clocks are specific to the PTP delay mechanism used in the network, either end-to-end (E2E) or peer-to-peer (P2P).

This document specifies a new transport for NTP to enable hardware timestamping on NICs which can timestamp only PTP messages and also take advantage of one-step E2E PTP unicast transparent clocks. It adds a new type-length-value (TLV) for PTP to contain NTP messages and adds a new extension field for NTP to provide clients and peers with the correction of their NTP requests from transparent clocks. The NTP broadcast mode is not supported.

The use of PTP messages requires that protocol rules of IEEE1588 [IEEE1588-2019] are followed. NTP over PTP does not require other PTP clocks to be present in the network. It does not disrupt their operation if they are present. If the network uses one-step E2E transparent clocks, NTP clients and peers using PTP for transport can reach the same or better accuracy as PTP clocks using PTP for synchronization. Hosts in a network can use PTP for synchronization in one domain and transport of NTP messages in another domain at the same time.

### 1.1. Comparison with PTP

The client-server mode of NTP, even with the PTP transport, has multiple advantages over PTP using multicast or unicast messaging:

- \* NTP is more secure. Existing security mechanisms specified for NTP like Network Time Security [RFC8915] are not impacted by the PTP transport. It is more difficult to secure PTP against delay attacks due to the sync message not being an immediate response to a client request. The PTP unicast mode allows an almost-infinite traffic amplification, which can be exploited for denial-of-service attacks and can only be limited by security mechanisms requiring client authentication.
- \* NTP is more resilient to failures. Each client can use multiple servers and detect failed sources in its source selection. In PTP a single hardware or software failure can disrupt the whole PTP domain. Multiple independent domains have to be used to handle any failure.
- \* NTP is better suited for synchronization in networks which do not have full on-path PTP support, or where timestamping errors do not have a symmetric distribution (e.g. due to sensitivity to network load). NTP does not assume network delay is constant and the rate of measurements in opposite directions is symmetric. It can filter the measurements more effectively and is not sensitive to asymmetrically distributed network delays and timestamping errors. PTP has to measure the offset and delay separately to enable multicast messaging, which is needed to reduce the transmit timestamping rate.
- \* NTP needs fewer messages to get the same number of timestamps. It uses less network bandwidth than PTP using unicast messaging.
- \* NTP provides clients with an estimate of the maximum error of the clock (root distance).

The disadvantage of NTP is transmit timestamping rate growing with the number of clients. A server which is limited by the hardware timestamping rate cannot provide a highly accurate time service to the same number of clients as with PTP using multicast messaging.

## 1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. PTP transport for NTP

A new TLV is defined for PTP to contain NTP messages in the client (3), server (4), and symmetric modes (1 and 2). Using other NTP modes in the TLV is not specified. Any transport specified for PTP that supports unicast messaging, and an IPv4 or IPv6 mapping, can be used for NTP over PTP.

The NTP TLV is an organization-specific TLV having the following fields (with octets in network order):

- \* type is 0x8000 (ORGANIZATION\_EXTENSION\_DO\_NOT\_PROPAGATE)
- \* lengthField is 8 + length of the NTP message
- \* organizationId is 00-00-5E (IANA OUI)
- \* organizationSubType is [[TBD]]
- \* dataField contains two zero octets for 32-bit alignment followed by the NTP message, which would normally be the UDP payload

The NTP TLV MUST be included in a unicast PTP event message. An event message is required to enable the PTP-specific hardware timestamping and corrections of transparent clocks. The PTP message MUST conform to PTP version 2 [IEEE1588-2008], PTP version 2.1 [IEEE1588-2019], or any future version of the PTP specification.

An NTP client or peer using the PTP transport sends NTP requests contained as the NTP TLV in PTP messages.

An NTP server or peer responding to an NTP request received over the PTP transport MUST send its response as the NTP TLV using the same PTP transport. The PTP message containing the NTP response MUST NOT be longer than the PTP message containing the NTP request. If the NTP response is expected to be used for synchronization (e.g. it is not an error message), the PTP message SHOULD have the same length as the message containing the request, using the PAD TLV (type 0x8008) if needed, to avoid an asymmetric delay in networks without full on-path PTP support.

The PTP version 2.1 [IEEE1588-2019] specification states that "A domain shall define the scope of PTP message communication, state, operations, data sets, and timescale. Within a PTP Network, a domain is identified by two attributes: domainNumber and sdoId.". In the context of NTP over PTP version 2.1, this means that the NTP servers, clients, and peers MUST verify that received PTP messages use the domainNumber and sdoId configured for use by NTP over PTP. The

domainNumber SHOULD be 123, and sdoId SHOULD be 0. This domainNumber 123 is not commonly used by PTP profiles, and so is less likely to interfere with any other PTP operation which might be running in the network.

If the UDP transport is used for PTP, the UDP source and destination port numbers SHOULD be the PTP event port (319). If the client implemented port randomization [RFC9109], requests and/or responses would not get a hardware receive timestamp due to the hardware filter matching only the PTP event port.

Any authenticator fields included in the NTP messages MUST be calculated only over the NTP message following the header of the NTP TLV. Other data in the PTP message (outside of the NTP TLV) are not protected. With the exception of the PTP correction field requiring special handling as described in the following section, the other PTP fields are used only for the transport of the NTP message and have no impact on security of NTP, similarly to the IP and UDP headers.

Receive and transmit timestamps contained in the NTP messages SHOULD NOT be adjusted for the beginning of the NTP data in the PTP message. To minimize the impact of different link speeds on accuracy in networks without full on-path PTP support, the transmit timestamp SHOULD correspond to the PTP message timestamp point (i.e. beginning of the first symbol after the Ethernet start of frame delimiter) and the receive timestamp SHOULD be transposed from the PTP message timestamp point to the ending of the reception (e.g. ending of the last symbol of the Ethernet frame check sequence).

### 3. Network Correction Extension Field

One-step E2E PTP transparent clocks modify the correction field in the header of the PTP event messages containing NTP messages. To be able to verify and apply the corrections to an NTP measurement, the client or peer needs to know the correction of both the request and response. The correction of the response is in the PTP header of the message itself. The correction of the request is provided by the server or other peer in a new NTP extension field included in the response.

The format of the Network Correction Extension Field is shown in Figure 1.

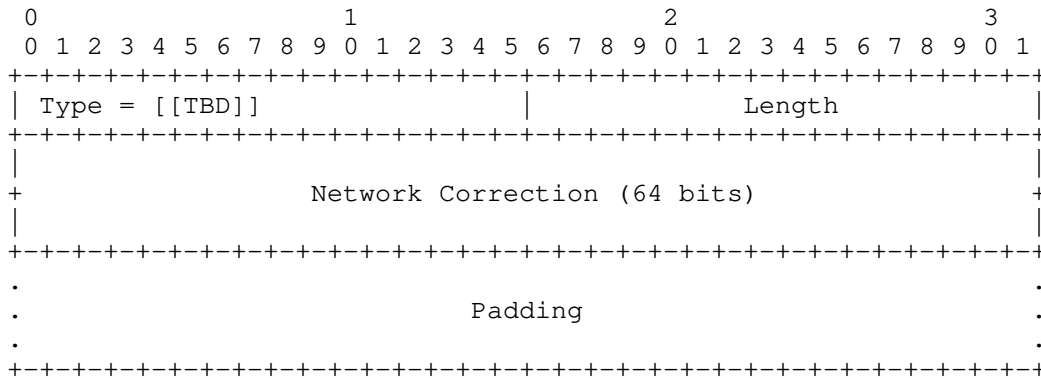


Figure 1: Format of Network Correction Extension Field

The length of the padding is the minimum required to make a valid extension field in the used version of NTP. In NTPv4 that is 16 octets to get a 28-octet extension field following RFC 7822 [RFC7822].

The Network Correction field in the extension field uses the 64-bit NTP timestamp format (resolution of about 1/4th of a nanosecond). The correction field in PTP header has a different format (64-bit nanoseconds + 16-bit fraction).

The value of the NTP network correction is the sum of PTP corrections provided by transparent clocks and the time it takes to receive the packet (i.e. packet length including the frame check sequence divided by the link speed).

The reason for not using the PTP correction alone is to avoid an asymmetric correction when the server and client, or peers, are connected to the network with different link speeds. The receive duration included in the NTP correction cancels out the transposition of PTP receive timestamp corresponding to the beginning of the reception to NTP receive timestamp corresponding to the end of the reception.

The Figure 2 shows the NTP timestamps, transmit/receive durations, and processing and queuing delays included in PTP corrections for an NTP exchange made over two PTP transparent clocks. The link speed is increasing on the network path from the client to the server. The propagation delays in cables are not shown.



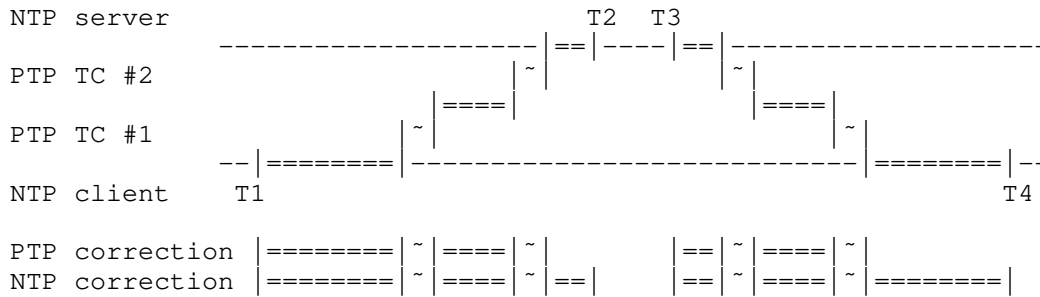


Figure 2: PTP vs NTP Correction

When an NTP server which supports the PTP transport receives an NTP request containing the Network Correction Extension Field, it SHOULD respond with the extension field providing the network correction of the client's request. The server MUST ignore the value of the network correction in the request.

An NTP client or peer which supports the PTP transport and is configured to use the network correction for the association SHOULD include the extension field in its NTP requests. In the case of a client, the correction value in the extension field SHOULD be always zero.

When the client or peer has the network correction of both the request and response, it can correct the measured NTP peer delay and offset:

$$* \text{ delta\_c} = \text{delta} - (\text{nc\_rs} + \text{nc\_rq} - \text{dur\_rs} - \text{dur\_rq}) * (1 - \text{freq\_tc})$$

$$* \text{ theta\_c} = \text{theta} + (\text{nc\_rs} - \text{nc\_rq}) / 2$$

where

- \* delta is the NTP peer delay from RFC 5905
- \* theta is the NTP offset from RFC 5905
- \* nc\_rq is the network correction of the request
- \* nc\_rs is the network correction of the response
- \* dur\_rq is the transmit duration of the request
- \* dur\_rs is the receive duration of the response

\* `freq_tc` is the maximum assumed frequency error of transparent clocks

The corrected delay (`delta_c`) and offset (`theta_c`) MUST NOT be accepted for synchronization if any of `delta_c`, `nc_rs`, and `nc_rq` is negative. This requirement limits the error caused by faulty transparent clocks and man-in-the-middle attacks.

Root delay (DELTA) MUST NOT be corrected to not make the maximum assumed error (root distance) dependent on accurate network corrections.

The scaling by the `freq_tc` constant (e.g. 100 ppm) is needed to make room for errors in corrections made by transparent clocks running faster than true time and avoid samples with larger corrections from getting a shorter delay than samples with smaller corrections, which would negatively impact their filtering and weighting.

The `dur_rq` and `dur_rs` values make the corrected peer delay correspond to a direct connection to the server. If they were not used, a perfectly corrected delay on a short network path would be too close to zero and frequently negative due to frequency offset between the client and server. Note that NTP peers and PTP clocks using the E2E delay mechanism are more sensitive to frequency offsets due to longer measurement intervals. If `dur_rq` is unknown, it MAY be assumed to be equal to `dur_rs`.

#### 4. Acknowledgements

The author would like to thank Doug Arnold, Rodney Cummings, and Martin Langer for their comments and suggestions.

#### 5. IANA Considerations

IANA is requested to allocate the following field in the NTP Extension Field Types Registry [RFC5905]:

Field Type	Meaning	Reference
[[TBD]]	Network correction	[[this memo]]

Table 1

IANA is requested to create a new registry "IANA PTP TLV Subtypes Registry" for entries having the following fields:

Subtype (REQUIRED) - integer in the range 0-0xFFFFFFFF

Description (REQUIRED)- short text description

Reference (REQUIRED) - reference to the document describing the IANA PTP TLV

Subtypes in the range 0x800000-0xFFFFFFFF are reserved for experimental and private use. They cannot be assigned by IANA.

The initial content of the registry is the following entry:

Subtype	Description	Reference
[[TBD]]	Network Time Protocol Message	[[this memo]]

Table 2

## 6. Implementation Status - RFC EDITOR: REMOVE BEFORE PUBLICATION

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

### 6.1. chrony

chrony (<https://chrony-project.org>) added experimental support for NTP over PTP in version 4.2. As the type of the NTP TLV, it uses 0x2023 from the experimental "do not propagate" range.

It was tested on Linux with the following network controllers, which have hardware timestamping limited to PTP packets:

Intel XL710 (i40e driver) - works

Intel X540-AT2 (ixgbe driver) - works

Intel 82576 (igb driver) - works

Broadcom BCM5720 (tg3 driver) - works

Broadcom BCM57810 (bnx2x driver) - does not timestamp unicast PTP packets

Solarflare SFC9250 (sfc driver) - works

The network correction was tested with the following switches which support operation as a one-step E2E PTP unicast transparent clock:

FS.COM IES3110-8TF-R - works

Juniper QFX5200-32C-32Q - works

## 7. Security Considerations

The PTP transport prevents NTP clients from randomizing their source port.

The corrections provided by PTP transparent clocks cannot be authenticated. Man-in-the-middle attackers can modify the correction field, but only corrections smaller than the measured delay are accepted by clients. The impact is comparable to the impact of delaying unmodified NTP messages.

## 8. References

### 8.1. Normative References

[IEEE1588-2019]

Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2019, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.", November 2019, <<https://www.ieee.org>>.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC7822] Mizrahi, T. and D. Mayer, "Network Time Protocol Version 4 (NTPv4) Extension Fields", RFC 7822, DOI 10.17487/RFC7822, March 2016, <<https://www.rfc-editor.org/info/rfc7822>>.

## 8.2. Informative References

- [IEEE1588-2008] Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.", July 2008, <<https://www.ieee.org>>.
- [RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/info/rfc8915>>.
- [RFC9109] Gont, F., Gont, G., and M. Lichvar, "Network Time Protocol Version 4: Port Randomization", RFC 9109, DOI 10.17487/RFC9109, August 2021, <<https://www.rfc-editor.org/info/rfc9109>>.

## Author's Address

Miroslav Lichvar  
Red Hat  
Email: [mlichvar@redhat.com](mailto:mlichvar@redhat.com)

Network Time Protocols  
Internet-Draft  
Intended status: Informational  
Expires: 3 February 2025

W. Ladd  
Akamai Technologies  
M. Dansarie  
Netnod  
2 August 2024

Roughtime  
draft-ietf-ntp-roughtime-11

Abstract

This document specifies Roughtime - a protocol that aims to achieve rough time synchronization even for clients without any idea of what time it is.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at  
<https://datatracker.ietf.org/doc/draft-ietf-ntp-roughtime/>.

Source for this draft and an issue tracker can be found at  
<https://github.com/wbl/roughtime-draft>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 February 2025.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Conventions and Definitions . . . . .	3
3. Protocol Overview . . . . .	3
4. The Guarantee . . . . .	4
5. Message Format . . . . .	4
5.1. Data types . . . . .	5
5.1.1. int32 . . . . .	5
5.1.2. uint32 . . . . .	5
5.1.3. uint64 . . . . .	5
5.1.4. Tag . . . . .	6
5.1.5. Timestamp . . . . .	6
5.2. Header . . . . .	6
6. Protocol Details . . . . .	7
6.1. Requests . . . . .	8
6.1.1. VER . . . . .	8
6.1.2. NONC . . . . .	8
6.1.3. SRV . . . . .	8
6.1.4. ZZZZ . . . . .	8
6.2. Responses . . . . .	9
6.2.1. SIG . . . . .	9
6.2.2. VER . . . . .	9
6.2.3. NONC . . . . .	9
6.2.4. PATH . . . . .	9
6.2.5. SREP . . . . .	10
6.2.6. CERT . . . . .	10
6.2.7. INDX . . . . .	10
6.3. The Merkle Tree . . . . .	11
6.3.1. Root Value Validity Check Algorithm . . . . .	11
6.4. Validity of Response . . . . .	12
7. Integration into NTP . . . . .	12
8. Grease . . . . .	12
9. RoughTime Clients . . . . .	13
9.1. Necessary configuration . . . . .	13
9.2. Measurement sequence . . . . .	13
9.3. Malfeasance reporting . . . . .	13
10. Security Considerations . . . . .	13
11. Privacy Considerations . . . . .	14

12. Operational Considerations . . . . .	14
13. IANA Considerations . . . . .	14
13.1. Service Name and Transport Protocol Port Number Registry . . . . .	14
13.2. Roughtime Version Registry . . . . .	15
13.3. Roughtime Tag Registry . . . . .	16
14. References . . . . .	17
14.1. Normative References . . . . .	17
14.2. Informative References . . . . .	18
Acknowledgments . . . . .	19
Authors' Addresses . . . . .	19

## 1. Introduction

Time synchronization is essential to Internet security as many security protocols and other applications require synchronization [RFC738]. Unfortunately, widely deployed protocols such as the Network Time Protocol (NTP) [RFC5905] lack essential security features, and even newer protocols like Network Time Security (NTS) [RFC8915] lack mechanisms to ensure that the servers behave correctly. Furthermore, clients may lack even a basic idea of the time, creating bootstrapping problems. Roughtime is intended to permit devices to obtain a rough idea of the current time from fairly static configuration consisting of a key and a server.

## 2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Protocol Overview

Roughtime is a protocol for rough time synchronization that enables clients to provide cryptographic proof of server malfeasance. It does so by having responses from servers include a signature over a value derived from a nonce in the client request. This provides cryptographic proof that the timestamp was issued after the server received the client's request. The derived value included in the server's response is the root of a Merkle tree which includes the hash of the client's nonce as the value of one of its leaf nodes. This enables the server to amortize the relatively costly signing operation over a number of client requests.



Single server mode: At its most basic level, Roughtime is a one round protocol in which a completely fresh client requests the current time and the server sends a signed response. The response includes a timestamp and a radius used to indicate the server's certainty about the reported time.

The server proves freshness of its response as follows. The client's request contains a nonce which the server incorporates into its signed response. The client can verify the server's signatures and - provided that the nonce has sufficient entropy - this proves that the signed response could only have been generated after the nonce.

#### 4. The Guarantee

A Roughtime server guarantees that a response to a query sent at  $t_1$ , received at  $t_2$ , and with timestamp  $t_3$  has been created between  $t_1$  and  $t_2$ . If  $t_3$  is not within that interval, a server inconsistency may be detected and used to impeach the server. The propagation of such a guarantee and its use for time synchronization is discussed in Section 7. No delay attacker may affect this: they may only expand the interval between  $t_1$  and  $t_2$ , or of course stop the measurement in the first place.

#### 5. Message Format

Roughtime messages are maps consisting of one or more (tag, value) pairs. They start with a header, which contains the number of pairs, the tags, and value offsets. The header is followed by a message values section which contains the values associated with the tags in the header. Messages MUST be formatted according to Figure 1 as described in the following sections.

Messages MAY be recursive, i.e. the value of a tag can itself be a Roughtime message.

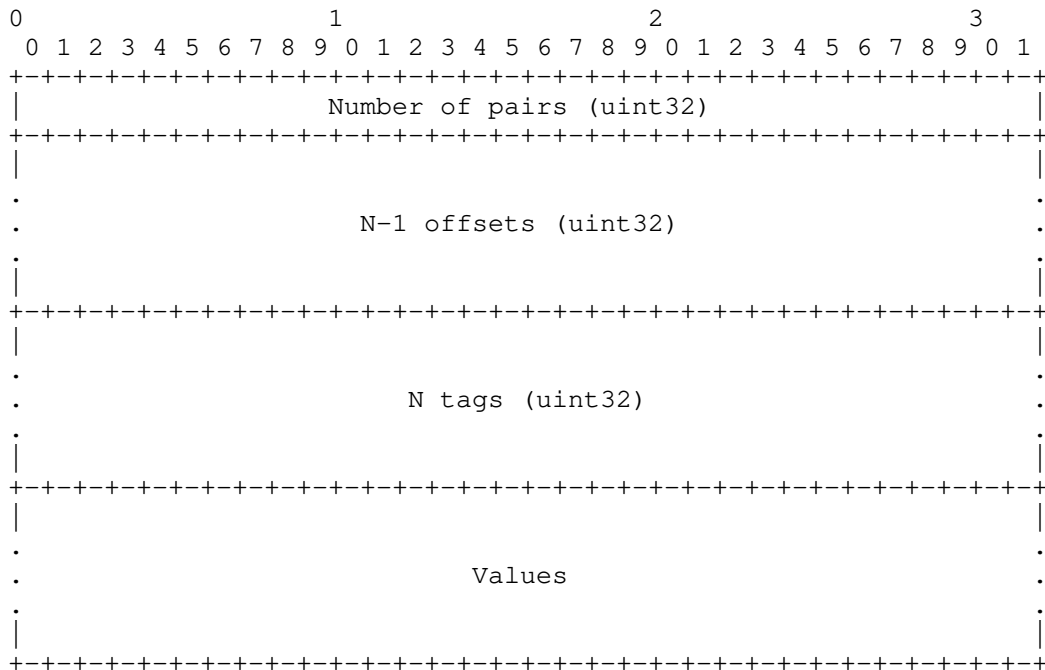


Figure 1: Roughtime Message

5.1. Data types

5.1.1. int32

An int32 is a 32 bit signed integer. It is serialized least significant byte first in sign-magnitude representation with the sign bit in the most significant bit. The negative zero value (0x80000000) MUST NOT be used and any message with it is syntactically invalid and MUST be ignored.

5.1.2. uint32

A uint32 is a 32 bit unsigned integer. It is serialized with the least significant byte first.

5.1.3. uint64

A uint64 is a 64 bit unsigned integer. It is serialized with the least significant byte first.

#### 5.1.4. Tag

Tags are used to identify values in Roughtime messages. A tag is a uint32 but can also be represented as a sequence of up to four ASCII characters [RFC20]. ASCII strings shorter than four characters can be unambiguously converted to tags by padding them with zero bytes. Tags MUST NOT contain any other bytes than capital letters (A-Z) or padding zero bytes. For example, the ASCII string "NONC" would correspond to the tag 0x434e4f4e and "ZZZZ" would correspond to 0x5a5a5a5a. Note that when encoded into a message the ASCII values will be in the natural bytewise order.

#### 5.1.5. Timestamp

A timestamp is a representation of UTC time as a uint64 count of seconds since 00:00:00 on 1 January 1970 (the Unix epoch), assuming every day has 86400 seconds. This is a constant offset from the NTP timestamp in seconds. Leap seconds do not have an unambiguous representation in a timestamp, and this has implications for the attainable accuracy and setting of the RADI tag.

#### 5.2. Header

All Roughtime messages start with a header. The first four bytes of the header is the uint32 number of tags  $N$ , and hence of (tag, value) pairs.

The following  $4*(N-1)$  bytes are offsets, each a uint32. The last  $4*N$  bytes in the header are tags. Offsets refer to the positions of the values in the message values section. All offsets MUST be multiples of four and placed in increasing order. The first post-header byte is at offset 0. The offset array is considered to have a not explicitly encoded value of 0 as its zeroth entry.

The value associated with the  $i$ th tag begins at  $\text{offset}[i]$  and ends at  $\text{offset}[i+1]-1$ , with the exception of the last value which ends at the end of the message. Values may have zero length. All lengths and offsets are in bytes.

Tags MUST be listed in the same order as the offsets of their values and MUST also be sorted in ascending order by numeric value. A tag MUST NOT appear more than once in a header.

6. Protocol Details

As described in Section 3, clients initiate time synchronization by sending requests containing a nonce to servers who send signed time responses in return. Roughtime packets can be sent between clients and servers either as UDP datagrams or via TCP streams. Servers SHOULD support the UDP transport mode and TCP mode.

A Roughtime packet MUST be formatted according to Figure 2 and as described here. The first field is a uint64 with the value 0x4d49544847554f52 ("ROUGHTIM" in ASCII). The second field is a uint32 and contains the length of the third field. The third and last field contains a Roughtime message as specified in Section 5.

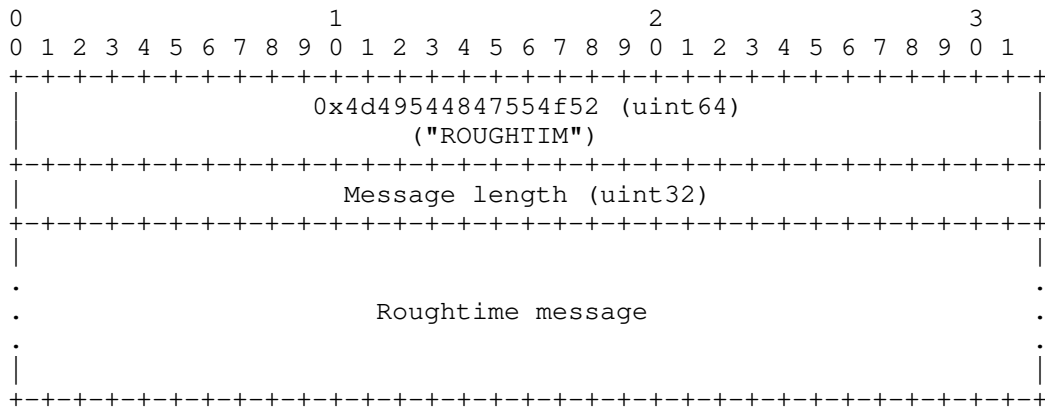


Figure 2: Roughtime packet

Roughtime request and response packets MUST be transmitted in a single datagram when the UDP transport mode is used. Setting the packet's don't fragment bit [RFC791] is OPTIONAL in IPv4 networks.

Multiple requests and responses can be exchanged over an established TCP connection. Clients MAY send multiple requests at once and servers MAY send responses out of order. The connection SHOULD be closed by the client when it has no more requests to send and has received all expected responses. Either side SHOULD close the connection in response to synchronization, format, implementation-defined timeouts, or other errors.

All requests and responses MUST contain the VER tag. It contains a list of one or more uint32 version numbers. The version of Roughtime specified by this memo has version number 1.

NOTE TO RFC EDITOR: remove this paragraph before publication. For testing drafts of this memo, a version number of 0x80000000 plus the draft number is used.

## 6.1. Requests

A request MUST contain the tags VER and NONC. It SHOULD include the tag SRV. Other tags SHOULD be ignored by the server. A future version of this protocol may mandate additional tags in the message and assign them semantic meaning.

The size of the request message SHOULD be at least 1024 bytes when the UDP transport mode is used. To attain this size the ZZZZ tag SHOULD be added to the message. Responding to requests shorter than 1024 bytes is OPTIONAL and servers MUST NOT send responses larger than the requests they are replying to.

### 6.1.1. VER

In a request, the VER tag contains a list of versions. The VER tag MUST include at least one Roughtime version supported by the client. The client MUST ensure that the version numbers and tags included in the request are not incompatible with each other or the packet contents.

The version numbers MUST NOT repeat.

### 6.1.2. NONC

The value of the NONC tag is a 32 byte nonce. It SHOULD be generated in a manner indistinguishable from random. BCP 106 contains specific guidelines regarding this [RFC4086].

### 6.1.3. SRV

The SRV tag is used by the client to indicate which long-term public key it expects to verify the response with. The value of the SRV tag is  $H(0xff || \text{public\_key})$  where `public_key` is the server's long-term, 32-byte Ed25519 public key and `H` is SHA512 truncated to the first 32 bytes.

### 6.1.4. ZZZZ

The ZZZZ tag is used to expand the response to the minimum required length. Its value MUST be a string of all zeros.

## 6.2. Responses

The server begins the request handling process with a set of long-term keys. It resolves which long-term key to use with the following procedure:

1. If the request contains a SRV tag, then the server looks up the long-term key indicated by the SRV value. If no such key exists, then the server MUST ignore the request.
2. If the request contains no SRV tag, but the server has just one long-term key, it SHOULD select that key. Otherwise, if the server has multiple long-term keys, then it MUST ignore the request.

A response MUST contain the tags SIG, VER, NONC, PATH, SREP, CERT, and INDX.

### 6.2.1. SIG

In general, a SIG tag value is a 64 byte Ed25519 signature [RFC8032] over a concatenation of a signature context ASCII string and the entire value of a tag. All context strings MUST include a terminating zero byte.

The SIG tag in the root of a response MUST be a signature over the SREP value using the public key contained in CERT. The context string MUST be "RoughTime v1 response signature".

### 6.2.2. VER

In a response, the VER tag MUST contain a single version number. It SHOULD be one of the version numbers supplied by the client in its request. The server MUST ensure that the version number corresponds with the rest of the packet contents.

### 6.2.3. NONC

The NONC tag MUST contain the nonce of the message being responded to.

### 6.2.4. PATH

The PATH tag value MUST be a multiple of 32 bytes long and represent a path of 32 byte hash values in the Merkle tree used to generate the ROOT value as described in a Section 6.3. In the case where a response is prepared for a single request and the Merkle tree contains only the root node, the size of PATH MUST be zero.

#### 6.2.5. SREP

The SREP tag contains a time response. Its value MUST be a Roughtime message with the tags ROOT, MIDP, and RADI.

The ROOT tag MUST contain a 32 byte value of a Merkle tree root as described in Section 6.3.

The MIDP tag value MUST be the timestamp of the moment of processing.

The RADI tag value MUST be a uint32 representing the server's estimate of the accuracy of MIDP in seconds. Servers MUST ensure that the true time is within (MIDP-RADI, MIDP+RADI) at the time they transmit the response message.

The value of the RADI tag MUST be at least 3 seconds. Otherwise leap seconds will impact the observed correctness of Roughtime servers.

#### 6.2.6. CERT

The CERT tag contains a public-key certificate signed with the server's long-term key. Its value is a Roughtime message with the tags DELE and SIG, where SIG is a signature over the DELE value. The context string used to generate SIG MUST be "RoughTime v1 delegation signature--".

The DELE tag contains a delegated public-key certificate used by the server to sign the SREP tag. Its value is a Roughtime message with the tags MINT, MAXT, and PUBK. The purpose of the DELE tag is to enable separation of a long-term public key from keys on devices exposed to the public Internet.

The MINT tag is the minimum timestamp for which the key in PUBK is trusted to sign responses. MIDP MUST be more than or equal to MINT for a response to be considered valid.

The MAXT tag is the maximum timestamp for which the key in PUBK is trusted to sign responses. MIDP MUST be less than or equal to MAXT for a response to be considered valid.

The PUBK tag contains a temporary 32 byte Ed25519 public key which is used to sign the SREP tag.

#### 6.2.7. INDX

The INDX tag value is a uint32 determining the position of NONC in the Merkle tree used to generate the ROOT value as described in Section 6.3.

### 6.3. The Merkle Tree

A Merkle tree is a binary tree where the value of each non-leaf node is a hash value derived from its two children. The root of the tree is thus dependent on all leaf nodes.

In Roughtime, each leaf node in the Merkle tree represents the nonce in one request. Leaf nodes are indexed left to right, beginning with zero.

The values of all nodes are calculated from the leaf nodes and up towards the root node using the first 32 bytes of the output of the SHA-512 hash algorithm [RFC6234]. For leaf nodes, the byte 0x00 is prepended to the nonce before applying the hash function. For all other nodes, the byte 0x01 is concatenated with first the left and then the right child node value before applying the hash function.

The value of the Merkle tree's root node is included in the ROOT tag of the response.

The index of a request's nonce node is included in the INDX tag of the response.

The values of all sibling nodes in the path between a request's nonce node and the root node are stored in the PATH tag so that the client can reconstruct and validate the value in the ROOT tag using its nonce. These values are each 32 bytes and are stored one after the other with no additional padding or structure. The order in which they are stored is described in the next section.

#### 6.3.1. Root Value Validity Check Algorithm

This section describes how to compute the root hash of the Merkle tree from the values in the tags PATH, INDX, and NONC. The bits of INDX are ordered from least to most significant.  $H(x)$  denotes the first 32 bytes of the SHA-512 hash digest of  $x$  and  $||$  denotes concatenation.

The algorithm maintains a current hash value. At initialization, hash is set to  $H(0x00 || \text{nonce})$ . When no more entries remain in PATH, the current hash is the hash of the Merkle tree. All remaining bits of INDX MUST be zero at that time. Otherwise, let node be the next 32 bytes in PATH. If the current bit in INDX is 0 then  $\text{hash} = H(0x01 || \text{node} || \text{hash})$ , else  $\text{hash} = H(0x01 || \text{hash} || \text{node})$ .

PATH is thus the siblings from the leaf to the root.



#### 6.4. Validity of Response

A client MUST check the following properties when it receives a response. We assume the long-term server public key is known to the client through other means.

The signature in CERT was made with the long-term key of the server.

The DELE timestamps and the MIDP value are consistent.

The INDX and PATH values prove NONC was included in the Merkle tree with value ROOT using the algorithm in Section 6.3.1.

The signature of SREP in SIG validates with the public key in DELE.

A response that passes these checks is said to be valid. Validity of a response does not prove the time is correct, but merely that the server signed it, and thus promises that it began to compute the signature at a time in the interval  $(MIDP-RADI, MIDP+RADI)$ .

#### 7. Integration into NTP

We assume that there is a bound PHI on the frequency error in the clock on the machine. Given a measurement taken at a local time  $t$ , we know the true time is in  $(t-\text{delta}-\text{sigma}, t-\text{delta}+\text{sigma})$ . After  $d$  seconds have elapsed we know the true time is within  $(t-\text{delta}-\text{sigma}-d\_PHI, t-\text{delta}+\text{sigma}+d\_PHI)$ . A simple and effective way to mix with NTP or PTP discipline of the clock is to trim the observed intervals in NTP to fit entirely within this window or reject measurements that fall to far outside. This assumes time has not been stepped. If the NTP process decides to step the time, it MUST use Roughtime to ensure the new truetime estimate that will be stepped to is consistent with the true time. Should this window become too large, another Roughtime measurement is called for. The definition of "too large" is implementation defined. Implementations MAY use other, more sophisticated means of adjusting the clock respecting Roughtime information. Other applications such as X.509 verification may wish to apply different rules.

#### 8. Grease

Servers SHOULD send back a fraction of responses that are syntactically invalid or contain invalid signatures as well as incorrect times. Clients MUST properly reject such responses. Servers MUST NOT send back responses with incorrect times and valid signatures. Either signature MAY be invalid for this application.

## 9. Roughtime Clients

### 9.1. Necessary configuration

To carry out a Roughtime measurement, a client must be equipped with a list of servers, a minimum of three of which are operational, not run by the same parties. It must also have a means of reporting to the provider of such a list, such as an OS vendor or software vendor, a failure report as described below.

### 9.2. Measurement sequence

The client randomly permutes three servers from the list, and sequentially queries them. The first probe uses a NONC that is randomly generated. The second query uses  $H(\text{resp} \parallel \text{rand})$  where `rand` is a random 32 byte value and `resp` is the entire response to the first probe. The third query uses  $H(\text{resp} \parallel \text{rand})$  for a different 32 byte value. If the times reported are consistent with the causal ordering, and the delay is within a system provided parameter, the measurement succeeds. If they are not consistent, there has been malfeasance and the client SHOULD store a report for evaluation, alert the operator, and make another measurement.

### 9.3. Malfeasance reporting

A malfeasance report is a JSON [RFC8259] object with keys "nonces", containing an array of the `rand` values as base64-encoded [RFC4648] strings, and "responses", containing an array of the responses as base64-encoded strings.

Malfeasance reports MAY be transported by any means to the relevant vendor or server operator for discussion. A malfeasance report is cryptographic proof that the responses arrived in that order, and can be used to demonstrate that at least one server sent the wrong time. The venues for sharing such reports and what to do about them are outside the scope of this document.

## 10. Security Considerations

Since the only supported signature scheme, Ed25519, is not quantum resistant, the Roughtime version described in this memo will not survive the advent of quantum computers.

Maintaining a list of trusted servers and adjudicating violations of the rules by servers is not discussed in this document and is essential for security. Roughtime clients MUST regularly update their view of which servers are trustworthy in order to benefit from the detection of misbehavior.

Validating timestamps made on different dates requires knowledge of leap seconds in order to calculate time intervals correctly.

Servers carry out a significant amount of computation in response to clients, and thus may experience vulnerability to denial of service attacks.

This protocol does not provide any confidentiality. Given the nature of timestamps, such impact is minor.

The compromise of a PUBK's private key, even past MAXT, is a problem as the private key can be used to sign invalid times that are in the range MINT to MAXT, and thus violate the good behavior guarantee of the server.

Servers MUST NOT send response packets larger than the request packets sent by clients, in order to prevent amplification attacks.

#### 11. Privacy Considerations

This protocol is designed to obscure all client identifiers. Servers necessarily have persistent long-term identities essential to enforcing correct behavior. Generating nonces in a nonrandom manner can cause leaks of private data or enable tracking of clients as they move between networks.

#### 12. Operational Considerations

It is expected that clients identify a server by its long-term public key. In multi-tenancy environments, where multiple servers may be listening on the same IP or port space, the protocol is designed so that the client indicates which server it expects to respond. This is done with the SRV tag.

#### 13. IANA Considerations

##### 13.1. Service Name and Transport Protocol Port Number Registry

IANA is requested to allocate the following entry in the Service Name and Transport Protocol Port Number Registry:

Service Name: Roughtime

Transport Protocol: tcp,udp

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Roughtime time synchronization

Reference: [[this memo]]

Port Number: [[TBD1]], selected by IANA from the User Port range

13.2. Roughtime Version Registry

IANA is requested to create a new registry entitled "Roughtime Version Registry". Entries shall have the following fields:

Version ID (REQUIRED): a 32-bit unsigned integer

Version name (REQUIRED): A short text string naming the version being identified.

Reference (REQUIRED): A reference to a relevant specification document.

The policy for allocation of new entries SHOULD be: IETF Review.

The initial contents of this registry shall be as follows:

Version ID	Version name	Reference
0x0	Reserved	[[this memo]]
0x1	Roughtime version 1	[[this memo]]
0x2-0x7fffffff	Unassigned	
0x80000000-0xffffffff	Reserved for Private or Experimental use	[[this memo]]

Table 1

### 13.3. Roughtime Tag Registry

IANA is requested to create a new registry entitled "Roughtime Tag Registry". Entries SHALL have the following fields:

Tag (REQUIRED): A 32-bit unsigned integer in hexadecimal format.

ASCII Representation (REQUIRED): The ASCII representation of the tag in accordance with Section 5.1.4 of this memo, if applicable.

Reference (REQUIRED): A reference to a relevant specification document.

The policy for allocation of new entries in this registry SHOULD be: Specification Required.

The initial contents of this registry SHALL be as follows:

Tag	ASCII Representation	Reference
0x00474953	SIG	[[this memo]]
0x00565253	SRV	[[this memo]]
0x00524556	VER	[[this memo]]
0x434e4f4e	NONC	[[this memo]]
0x454c4544	DELE	[[this memo]]
0x48544150	PATH	[[this memo]]
0x49444152	RADI	[[this memo]]
0x4b425550	PUBK	[[this memo]]
0x5044494d	MIDP	[[this memo]]
0x50455253	SREP	[[this memo]]
0x544e494d	MINT	[[this memo]]
0x544f4f52	ROOT	[[this memo]]
0x54524543	CERT	[[this memo]]
0x5458414d	MAXT	[[this memo]]
0x58444e49	INDX	[[this memo]]
0x5a5a5a5a	ZZZZ	[[this memo]]

Table 2

## 14. References

### 14.1. Normative References

- [RFC20] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <<https://www.rfc-editor.org/rfc/rfc20>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/rfc/rfc4086>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/rfc/rfc4648>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/rfc/rfc6234>>.
- [RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/rfc/rfc791>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/rfc/rfc8032>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/rfc/rfc8259>>.

#### 14.2. Informative References

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/rfc/rfc5905>>.
- [RFC738] Harrenstien, K., "Time server", RFC 738, DOI 10.17487/RFC0738, October 1977, <<https://www.rfc-editor.org/rfc/rfc738>>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/rfc/rfc8915>>.

#### Acknowledgments

Aanchal Malhotra and Adam Langley authored early drafts of this memo. Thomas Peterson corrected multiple nits. Peter Lauthberg, Tal Mizrahi, Ragnar Sundblad, Kristof Teichel, and the other members of the NTP working group contributed comments and suggestions.

#### Authors' Addresses

Watson Ladd  
Akamai Technologies  
Email: [watsonbladd@gmail.com](mailto:watsonbladd@gmail.com)

Marcus Dansarie  
Netnod  
Email: [marcus@dansarie.se](mailto:marcus@dansarie.se)  
URI: <https://orcid.org/0000-0001-9246-0263>



Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: 18 March 2022

W. Ladd  
Cloudflare  
M. Dansarie  
September 2021

Roughtime Ecosystem  
draft-ietf-ntp-roughtime-ecosystem-01

Abstract

This document specifies the roles of Roughtime validators, clients, and servers in providing a ecosystem for secure time.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 March 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Chaining in roughtime . . . . .	2
3. Impeachment . . . . .	2
4. Serialization of chains . . . . .	3
5. Submission API . . . . .	3
6. Viewing Reports . . . . .	3
7. Trust Anchors and Policies . . . . .	3
8. Normative References . . . . .	3
Authors' Addresses . . . . .	3

## 1. Introduction

The Roughtime protocol enables servers to provide cryptographic proof of the times requests were made. This enables clients to expose cheating by servers. This document describes how these proofs are serialized and verified, as well as APIs to access and submit reports of malfeasance in an automated manner.

## 2. Chaining in roughtime

Two responses are chained if the NONC field of the second is SHA-512(blinder || first) where blinder is a 64 byte value. Blinder MUST be generated uniformly at random to prevent tracking. The first response is serialized as a roughtime message. The first response is chained to the second.

A chain is a sequence of messages where each message is chained to the one before. Every contiguous subsequence of a chain is a chain.

## 3. Impeachment

For each index  $i$ , let  $m_i$  denote the timestamp of the response,  $r_i$  the radius around it. Then we have  $m_i - r_i$  the earliest actual time at which the response could have been generated, and  $m_i + r_i$  the latest actual time at which the response could have been generated.

If all requests are generated honestly  $m_i + r_i < m_{i+j} - r_{i+j}$  holds for all indices  $i$  and positive numbers  $j$ . A failure of this relation to hold demonstrates that at least one of the responses was generated incorrectly.

The more distinct servers and responses that are mutually consistent except for the questionable response, the more likely a failure of the generator of the erroneous response is.

#### 4. Serialization of chains

TODO

#### 5. Submission API

#### 6. Viewing Reports

#### 7. Trust Anchors and Policies

A trust anchor is any distributor of a list of trusted servers. It is RECOMMENDED that trust anchors subscribe to a common public forum where evidence of malfeasance may be shared and discussed. Trust anchors SHOULD subscribe to a zero-tolerance policy: any generation of incorrect timestamps will result in removal. To enable this trust anchors SHOULD list a wide variety of servers so the removal of a server does not result in operational issues for clients. Clients SHOULD attempt to detect malfeasance and report it as discussed in this document.

Because only a single RoughTime server is required for successful synchronization, RoughTime does not have the incentive problems that have prevented effective enforcement of discipline on the web PKI.

#### 8. Normative References

[I-D.ietf-ntp-roughTime]

Malhotra, A., Langley, A., Ladd, W., and M. Dansarie,  
"RoughTime", Work in Progress, Internet-Draft, draft-ietf-  
ntp-roughTime-05, 24 May 2021,  
<<https://www.ietf.org/archive/id/draft-ietf-ntp-roughTime-05.txt>>.

#### Authors' Addresses

Watson Ladd  
Cloudflare  
101 Townsend St  
San Francisco,  
United States of America

Email: [watsonbladd@gmail.com](mailto:watsonbladd@gmail.com)

Marcus Dansarie  
Sweden

Email: [marcus@dansarie.se](mailto:marcus@dansarie.se)

URI: <https://orcid.org/0000-0001-9246-0263>

ntp  
Internet-Draft  
Updates: 5905, 5906, 8573, 7822, 7821 (if  
approved)  
Intended status: Standards Track  
Expires: 21 February 2025

R. Salz  
Akamai Technologies  
20 August 2024

Updating the NTP Registries  
draft-ietf-ntp-update-registries-16

Abstract

The Network Time Protocol (NTP) and Network Time Security (NTS) documents define a number of assigned number registries, collectively called the NTP registries.

Some registries have wrong values, some registries do not follow current common practice, and some are just right. For the sake of completeness, this document reviews all NTP and NTS registries, and makes updates where necessary.

This document updates RFC 5905, RFC 5906, RFC 8573, RFC 7822, and RFC 7821.

Notes

This note is to be removed before publishing as an RFC.

This document is a product of the NTP Working Group (<https://dt.ietf.org/wg/ntp>). Source for this draft and an issue tracker can be found at <https://github.com/richsalz/draft-rsalz-update-registries>.

RFC Editor: Please update 'this RFC' to refer to this document, once its RFC number is known, through the document.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 February 2025.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- 1. Introduction . . . . . 2
- 2. Existing Registries . . . . . 3
  - 2.1. Reference ID, Kiss-o'-Death . . . . . 3
  - 2.2. Extension Field Types . . . . . 3
  - 2.3. Network Time Security Registries . . . . . 4
- 3. Updated Registries . . . . . 5
  - 3.1. Guidance to Designated Experts . . . . . 5
- 4. IANA Considerations . . . . . 5
  - 4.1. NTP Reference Identifier Codes . . . . . 5
  - 4.2. NTP Kiss-o'-Death Codes . . . . . 6
  - 4.3. NTP Extension Field Types . . . . . 6
- 5. Security Considerations . . . . . 10
- 6. Acknowledgements . . . . . 10
- 7. Normative References . . . . . 10
- Author's Address . . . . . 11

1. Introduction

The Network Time Protocol (NTP) and Network Time Security (NTS) documents define a number of assigned number registries, collectively called the NTP registries. The NTP registries can all be found at <https://www.iana.org/assignments/ntp-parameters/ntp-parameters.xhtml> (<https://www.iana.org/assignments/ntp-parameters/ntp-parameters.xhtml>) and the NTS registries can all be found at <https://www.iana.org/assignments/nts/nts.xhtml>

(<https://www.iana.org/assignments/nts/nts.xhtml>).

Some registries have wrong values, some registries do not follow current common practice, and some are just right. For the sake of completeness, this document reviews all NTP and NTS registries, and makes updates where necessary.

The bulk of this document can be divided into two parts:

- \* First, each registry, its defining document, and a summary of its syntax is defined.
- \* Second, the revised format and entries for each registry that is being modified is specified.

## 2. Existing Registries

This section describes the registries and the rules for them. It is intended to be a short summary of the syntax and registration requirements for each registry. The semantics and protocol processing rules for each registry -- that is, how an implementation acts when sending or receiving any of the fields -- are not described here.

### 2.1. Reference ID, Kiss-o'-Death

[RFC5905] defined two registries; the Reference ID in Section 7.3, and the Kiss-o'-Death in Section 7.4. Both of these are allowed to be four ASCII characters; padded on the right with all-bits-zero if necessary. Entries that start with 0x58, the ASCII letter uppercase X, are reserved for Private or Experimental Use. Both registries are first-come first-served. The formal request to define the registries is in [RFC5905], Section 16.

### 2.2. Extension Field Types

[RFC5905], Section 7.5 defined the on-the-wire format of extension fields but did not create a registry for them.

[RFC5906], Section 13 mentioned the Extension Field Types registry, and defined it indirectly by defining 30 extensions (10 each for request, response, and error response). It did not provide a formal definition of the columns in the registry. [RFC5906], Section 10 splits the Field Type into four subfields, only for use within the Autokey extensions.

[RFC7821] added a new entry, Checksum Complement, to the Extension Field Types registry.

[RFC7822] clarified the processing rules for Extension Field Types, particularly around the interaction with the Message Authentication Code (MAC) field. NTPv4 packets may contain a MAC that appears where one would expect the next extension field header.

[RFC8573] changed the cryptography used in the MAC field.

[RFC8915] added four new entries to the Extension Field Types registry.

The following problems exist with the current registry:

- \* Many of the entries in the Extension Field Types registry have swapped some of the nibbles; 0x1234 is listed as 0x1432 for example. This was due to documentation errors with the original implementation of Autokey. This document marks the erroneous values as reserved, in case there is an implementation that used the registered values instead of what the original implementation used. Applications that might have used those values would have realized that they did not interoperate with the dominant (if not only) implementation at the time. Marking the values as reserved ensures that any such applications would still be able to work as-is.
- \* Some values were mistakenly re-used.

### 2.3. Network Time Security Registries

[RFC8915] defines the NTS protocol. Its registries are listed here for completeness, but no changes to them are specified in this document.

Sections 7.1 through 7.5 (inclusive) added entries to existing registries.

Section 7.6 created a new registry, NTS Key Establishment Record Types, that partitions the assigned numbers into three different registration policies: IETF Review, Specification Required, and Private or Experimental Use.

Section 7.7 created a new registry, NTS Next Protocols, that similarly partitions the assigned numbers.

Section 7.8 created two new registries, NTS Error Codes and NTS Warning Codes. Both registries are also partitioned the same way.



### 3. Updated Registries

The following general guidelines apply to all registries updated here:

- \* Every registry reserves a partition for Private or Experimental Use.
- \* Entries with ASCII fields are now limited to uppercase letters or digits; fields starting with 0x58, the uppercase letter "X", are reserved for Private or Experimental Use.
- \* The policy for every registry is now Specification Required, as defined in [RFC8126], Section 4.6.

The IESG is requested to choose three designated experts, with two being required to approve a registry change. Guidance for such experts is given below.

Each entry described in the sub-sections below is intended to completely replace the existing entry with the same name.

#### 3.1. Guidance to Designated Experts

The designated experts (DE) should be familiar with [RFC8126], particularly Section 5. As that reference suggests, the DE should ascertain the existence of a suitable specification, and verify that it is publicly available. The DE is also expected to check the clarity of purpose and use of the requested code points.

In addition, the DE is expected to be familiar with this document, specifically the history documented here.

### 4. IANA Considerations

#### 4.1. NTP Reference Identifier Codes

The registration procedure is changed to Specification Required.

The Note is changed to read as follows:

- \* Codes beginning with the character "X" are reserved for experimentation and development. IANA cannot assign them.

The columns are defined as follows:

- \* ID (required): a four-byte value padded on the right with all-bits-zero. Each byte other than padding must be an ASCII uppercase letter or digits.
- \* Clock source (required): A brief text description of the ID.
- \* Reference (required): the publication defining the ID.

The existing entries are left unchanged.

#### 4.2. NTP Kiss-o'-Death Codes

The registration procedure is changed to Specification Required.

The Note is changed to read as follows:

- \* Codes beginning with the character "X" are reserved for experimentation and development. IANA cannot assign them.

The columns are defined as follows:

- \* ID (required): a four-byte value padded on the right with all-bits-zero. Each byte other than padding must be an ASCII uppercase letter or digits.
- \* Meaning source (required): A brief text description of the ID.
- \* Reference (required): the publication defining the ID.

The existing entries are left unchanged.

#### 4.3. NTP Extension Field Types

The registration procedure is changed to Specification Required.

The reference [RFC5906] should be added, if possible.

The following two Notes are added:

- \* Field Types in the range 0xF000 through 0xFFFF, inclusive, are reserved for experimentation and development. IANA cannot assign them. Both NTS Cookie and Autokey Message Request have the same Field Type; in practice this is not a problem as the field semantics will be determined by other parts of the message.

- \* The "Reserved for historic reasons" is for differences between the original documentation and implementation of Autokey and marks the erroneous values as reserved, in case there is an implementation that used the registered values instead of what the original implementation used.

The columns are defined as follows:

- \* Field Type (required): A two-byte value in hexadecimal.
- \* Meaning (required): A brief text description of the field type.
- \* Reference (required): the publication defining the field type.

The table is replaced with the following entries. IANA is requested to replace "This RFC" with the actual RFC number once assigned.

Field Type	Meaning	Reference
0x0000	Crypto-NAK; authentication failure	RFC 5905
0x0002	Reserved for historic reasons	This RFC
0x0102	Reserved for historic reasons	This RFC
0x0104	Unique Identifier	RFC 8915, Section 5.3
0x0200	No-Operation Request	RFC 5906
0x0201	Association Message Request	RFC 5906
0x0202	Certificate Message Request	RFC 5906
0x0203	Cookie Message Request	RFC 5906
0x0204	Autokey Message Request	RFC 5906
0x0204	NTS Cookie	RFC 8915, Section 5.4
0x0205	Leapseconds Message Request	RFC 5906
0x0206	Sign Message Request	RFC 5906
0x0207	IFF Identity Message Request	RFC 5906

0x0208	GQ Identity Message Request	RFC 5906
0x0209	MV Identity Message Request	RFC 5906
0x0302	Reserved for historic reasons	This RFC
0x0304	NTS Cookie Placeholder	RFC 8915, Section 5.5
0x0402	Reserved for historic reasons	This RFC
0x0404	NTS Authenticator and Encrypted Extension Fields	RFC 8915, Section 5.6
0x0502	Reserved for historic reasons	This RFC
0x0602	Reserved for historic reasons	This RFC
0x0702	Reserved for historic reasons	This RFC
0x0902	Reserved for historic reasons	This RFC
0x2005	UDP Checksum Complement	RFC 7821
0x8002	Reserved for historic reasons	This RFC
0x8102	Reserved for historic reasons	This RFC
0x8200	No-Operation Response	RFC 5906
0x8201	Association Message Response	RFC 5906
0x8202	Certificate Message Response	RFC 5906
0x8203	Cookie Message Response	RFC 5906
0x8204	Autokey Message Response	RFC 5906
0x8205	Leapseconds Message Response	RFC 5906
0x8206	Sign Message Response	RFC 5906
0x8207	IFF Identity Message Response	RFC 5906
0x8208	GQ Identity Message Response	RFC 5906
0x8209	MV Identity Message Response	RFC 5906

0x8302	Reserved for historic reasons	This RFC
0x8402	Reserved for historic reasons	This RFC
0x8502	Reserved for historic reasons	This RFC
0x8602	Reserved for historic reasons	This RFC
0x8702	Reserved for historic reasons	This RFC
0x8802	Reserved for historic reasons	This RFC
0x8902	Reserved for historic reasons	This RFC
0xC002	Reserved for historic reasons	This RFC
0xC102	Reserved for historic reasons	This RFC
0xC200	No-Operation Error Response	RFC 5906
0xC201	Association Message Error Response	RFC 5906
0xC202	Certificate Message Error Response	RFC 5906
0xC203	Cookie Message Error Response	RFC 5906
0xC204	Autokey Message Error Response	RFC 5906
0xC205	Leapseconds Message Error Response	RFC 5906
0xC206	Sign Message Error Response	RFC 5906
0xC207	IFF Identity Message Error Response	RFC 5906
0xC208	GQ Identity Message Error Response	RFC 5906
0xC209	MV Identity Message Error Response	RFC 5906
0xC302	Reserved for historic reasons	This RFC
0xC402	Reserved for historic reasons	This RFC
0xC502	Reserved for historic reasons	This RFC
0xC602	Reserved for historic reasons	This RFC
0xC702	Reserved for historic reasons	This RFC

0xC802	Reserved for historic reasons	This RFC
0xC902	Reserved for historic reasons	This RFC
0xF000- 0xFFFF	Reserved for Experimental Use	This RFC

Table 1

## 5. Security Considerations

This document adds no new security considerations, as they are defined in the document that defines the extension. See the References column of the appropriate table.

## 6. Acknowledgements

The members of the NTP Working Group helped a great deal. Notable contributors include:

- \* Miroslav Lichvar, Red Hat
- \* Daniel Franke, formerly at Akamai Technologies
- \* Danny Mayer, Network Time Foundation
- \* Michelle Cotton, formerly at IANA
- \* Tamme Dittrich, Tweede Golf

## 7. Normative References

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/rfc/rfc5905>>.
- [RFC5906] Haberman, B., Ed. and D. Mills, "Network Time Protocol Version 4: Autokey Specification", RFC 5906, DOI 10.17487/RFC5906, June 2010, <<https://www.rfc-editor.org/rfc/rfc5906>>.
- [RFC7821] Mizrahi, T., "UDP Checksum Complement in the Network Time Protocol (NTP)", RFC 7821, DOI 10.17487/RFC7821, March 2016, <<https://www.rfc-editor.org/rfc/rfc7821>>.

- [RFC7822] Mizrahi, T. and D. Mayer, "Network Time Protocol Version 4 (NTPv4) Extension Fields", RFC 7822, DOI 10.17487/RFC7822, March 2016, <<https://www.rfc-editor.org/rfc/rfc7822>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/rfc/rfc8126>>.
- [RFC8573] Malhotra, A. and S. Goldberg, "Message Authentication Code for the Network Time Protocol", RFC 8573, DOI 10.17487/RFC8573, June 2019, <<https://www.rfc-editor.org/rfc/rfc8573>>.
- [RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/rfc/rfc8915>>.

## Author's Address

Rich Salz  
Akamai Technologies  
Email: [rsalz@akamai.com](mailto:rsalz@akamai.com)

TICTOC Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 24 January 2025

D.A. Arnold  
Meinberg-USA  
H.G. Gerstung  
Meinberg  
23 July 2024

Enterprise Profile for the Precision Time Protocol With Mixed Multicast  
and Unicast messages  
draft-ietf-tictoc-ntp-enterprise-profile-28

Abstract

This document describes a Precision Time Protocol (PTP) Profile IEEE 1588-2019 [IEEE1588] for use in an IPv4 or IPv6 Enterprise information system environment. The PTP Profile uses the End-to-End delay measurement mechanism, allows both multicast and unicast Delay Request and Delay Response messages.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 24 January 2025.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.



This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction	2
2. Requirements Language	4
3. Technical Terms	4
4. Problem Statement	6
5. Network Technology	7
6. Time Transfer and Delay Measurement	8
7. Default Message Rates	9
8. Requirements for TimeTransmitter Clocks	9
9. Requirements for TimeReceiver Clocks	10
10. Requirements for Transparent Clocks	11
11. Requirements for Boundary Clocks	11
12. Management and Signaling Messages	11
13. Forbidden PTP Options	11
14. Interoperation with IEEE 1588 Default Profile	11
15. Profile Identification	12
16. Acknowledgements	12
17. IANA Considerations	12
18. Security Considerations	12
19. References	13
19.1. Normative References	13
19.2. Informative References	14
Authors' Addresses	14

## 1. Introduction

The Precision Time Protocol ("PTP"), standardized in IEEE 1588, has been designed in its first version (IEEE 1588-2002) with the goal to minimize configuration on the participating nodes. Network communication was based solely on multicast messages, which unlike NTP did not require that a receiving node in IEEE 1588-2019 [IEEE1588] need to know the identity of the time sources in the network. This document describes clock roles and PTP Port states using the optional alternative terms *timeTransmitter*, instead of *master*, and *timeReceiver*, instead of *slave*, as defined in the IEEE 1588g [IEEE1588g] amendment to IEEE 1588-2019 [IEEE1588].

The "Best TimeTransmitter Clock Algorithm" (IEEE 1588-2019 [IEEE1588] Subclause 9.3), a mechanism that all participating PTP nodes MUST follow, set up strict rules for all members of a PTP domain to determine which node MUST be the active reference time source (Grandmaster). Although the multicast communication model has advantages in smaller networks, it complicated the application of PTP in larger networks, for example in environments like IP based telecommunication networks or financial data centers. It is considered inefficient that, even if the content of a message applies only to one receiver, it is forwarded by the underlying network (IP) to all nodes, requiring them to spend network bandwidth and other resources, such as CPU cycles, to drop the message.

The third edition of the standard (IEEE 1588-2019) defines PTPv2.1 and includes the possibility to use unicast communication between the PTP nodes in order to overcome the limitation of using multicast messages for the bi-directional information exchange between PTP nodes. The unicast approach avoided that. In PTP domains with a lot of nodes, devices had to throw away most of the received multicast messages because they carried information for some other node. The percent of PTP message that are discarded as irrelevant to the receiving node can exceed 99% (Estrela and Bonebakker [Estrela\_and\_Bonebakker]).

PTPv2.1 also includes PTP Profiles (IEEE 1588-2019 [IEEE1588] subclause 20.3). This construct allows organizations to specify selections of attribute values and optional features, simplifying the configuration of PTP nodes for a specific application. Instead of having to go through all possible parameters and configuration options and individually set them up, selecting a PTP Profile on a PTP node will set all the parameters that are specified in the PTP Profile to a defined value. If a PTP Profile definition allows multiple values for a parameter, selection of the PTP Profile will set the profile-specific default value for this parameter. Parameters not allowing multiple values are set to the value defined in the PTP Profile. Many PTP features and functions are optional, and a PTP Profile should also define which optional features of PTP are required, permitted, and prohibited. It is possible to extend the PTP standard with a PTP Profile by using the TLV mechanism of PTP (see IEEE 1588-2019 [IEEE1588] subclause 13.4), defining an optional Best TimeTransmitter Clock Algorithm and a few other ways. PTP has its own management protocol (defined in IEEE 1588-2019 [IEEE1588] subclause 15.2) but allows a PTP Profile to specify an alternative management mechanism, for example NETCONF.

In this document the term PTP Port refers to a logical access point of a PTP instantiation for PTP communication in a network.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Technical Terms

- \* **Acceptable TimeTransmitter Table:** A PTP timeReceiver Clock may maintain a list of timeTransmitters which it is willing to synchronize to.
- \* **Alternate timeTransmitter:** A PTP timeTransmitter Clock, which is not the Best timeTransmitter, may act as a timeTransmitter with the Alternate timeTransmitter flag set on the messages it sends.
- \* **Announce message:** Contains the timeTransmitter Clock properties of a timeTransmitter Clock. Used to determine the Best TimeTransmitter.
- \* **Best timeTransmitter:** A clock with a PTP Port in the timeTransmitter state, operating as the Grandmaster of a PTP domain.
- \* **Best TimeTransmitter Clock Algorithm:** A method for determining which state a PTP Port of a PTP clock should be in. The state decisions lead to the formation of a clock spanning tree for a PTP domain.
- \* **Boundary Clock:** A device with more than one PTP Port. Generally Boundary Clocks will have one PTP Port in timeReceiver state to receive timing and other PTP Ports in timeTransmitter state to re-distribute the timing.
- \* **Clock Identity:** In IEEE 1588-2019 this is a 64-bit number assigned to each PTP clock which MUST be globally unique. Often it is derived from the Ethernet MAC address.
- \* **Domain:** Every PTP message contains a domain number. Domains are treated as separate PTP systems in the network. Clocks, however, can combine the timing information derived from multiple domains.

- \* End-to-End delay measurement mechanism: A network delay measurement mechanism in PTP facilitated by an exchange of messages between a timeTransmitter Clock and a timeReceiver Clock. These messages might traverse Transparent Clocks and PTP unaware switches. This mechanism might not work properly if the Sync and Delay Request messages traverse different network paths.
- \* Grandmaster: the timeTransmitter Clock that is currently acting as the reference time source of the PTP domain
- \* IEEE 1588: The timing and synchronization standard which defines PTP, and describes the node, system, and communication properties necessary to support PTP.
- \* TimeTransmitter Clock: a clock with at least one PTP Port in the timeTransmitter state.
- \* NTP: Network Time Protocol, defined by RFC 5905, see RFC 5905 [RFC5905]
- \* Ordinary Clock: A clock that has a single Precision Time Protocol PTP Port in a domain and maintains the timescale used in the domain. It may serve as a timeTransmitter Clock, or be a timeReceiver Clock.
- \* Peer-to-Peer delay measurement mechanism: A network delay measurement mechanism in PTP facilitated by an exchange of messages over the link between adjacent devices in a network. This mechanism might not work properly unless all devices in the network support PTP and the Peer-to-peer measurement mechanism.
- \* Preferred timeTransmitter: A device intended to act primarily as the Grandmaster of a PTP system, or as a back up to a Grandmaster.
- \* PTP: The Precision Time Protocol: The timing and synchronization protocol defined by IEEE 1588.
- \* PTP Port: An interface of a PTP clock with the network. Note that there may be multiple PTP Ports running on one physical interface, for example, multiple unicast timeReceivers which talk to several Grandmaster Clocks in different PTP Domains.
- \* PTP Profile: A set of constraints on the options and features of PTP, designed to optimize PTP for a specific use case or industry. The profile specifies what is required, allowed and forbidden among options and attribute values of PTP.

- \* PTPv2.1: Refers specifically to the version of PTP defined by IEEE 1588-2019.
- \* Rogue timeTransmitter: A clock with a PTP Port in the timeTransmitter state, even though it should not be in the timeTransmitter state according to the Best TimeTransmitter Clock Algorithm, and does not set the Alternate timeTransmitter flag.
- \* TimeReceiver Clock: a clock with at least one PTP Port in the timeReceiver state, and no PTP Ports in the timeTransmitter state.
- \* TimeReceiver Only clock: An Ordinary Clock which cannot become a timeTransmitter Clock.
- \* TLV: Type Length Value, a mechanism for extending messages in networked communications.
- \* Transparent Clock. A device that measures the time taken for a PTP event message to transit the device and then updates the message with a correction for this transit time.
- \* Unicast Discovery: A mechanism for PTP timeReceivers to establish a unicast communication with PTP timeTransmitters using a configured table of timeTransmitter IP addresses and Unicast Message Negotiation.
- \* Unicast Negotiation: A mechanism in PTP for timeReceiver Clocks to negotiate unicast Sync, Announce and Delay Request message transmission rates from timeTransmitters.

#### 4. Problem Statement

This document describes how PTP can be applied to work in large enterprise networks. See ISPCS [RFC2026] for information on IETF applicability statements. Such large networks are deployed, for example, in financial corporations. It is becoming increasingly common in such networks to perform distributed time tagged measurements, such as one-way packet latencies and cumulative delays on software systems spread across multiple computers. Furthermore, there is often a desire to check the age of information time tagged by a different machine. To perform these measurements, it is necessary to deliver a common precise time to multiple devices on a network. Accuracy currently required in the Financial Industry range from 100 microseconds to 1 nanoseconds to the Grandmaster. This PTP Profile does not specify timing performance requirements, but such requirements explain why the needs cannot always be met by NTP, as commonly implemented. Such accuracy cannot usually be achieved with a traditional time transfer such as NTP, without adding non-standard

customizations such as on-path support, similar to what is done in PTP with Transparent Clocks and Boundary Clocks. Such PTP support is commonly available in switches and routers, and many such devices have already been deployed in networks. Because PTP has a complex range of features and options it is necessary to create a PTP Profile for enterprise networks to achieve interoperability between equipment manufactured by different vendors.

Although enterprise networks can be large, it is becoming increasingly common to deploy multicast protocols, even across multiple subnets. For this reason, it is desired to make use of multicast whenever the information going to many destinations is the same. It is also advantageous to send information which is only relevant to one device as a unicast message. The latter can be essential as the number of PTP timeReceivers becomes hundreds or thousands.

PTP devices operating in these networks need to be robust. This includes the ability to ignore PTP messages which can be identified as improper, and to have redundant sources of time.

Interoperability among independent implementations of this PTP Profile has been demonstrated at the ISPCS Plugfest ISPCS [ISPCS].

## 5. Network Technology

This PTP Profile MUST operate only in networks characterized by UDP RFC 768 [RFC0768] over either IPv4 RFC 791 [RFC0791] or IPv6 RFC 8200 [RFC8200], as described by Annexes C and D in IEEE 1588 [IEEE1588] respectively. A network node MAY include multiple PTP instances running simultaneously. IPv4 and IPv6 instances in the same network node MUST operate in different PTP Domains. PTP Clocks which communicate using IPv4 can transfer time to PTP Clocks using IPv6, or the reverse, if and only if, there is a network node which simultaneously communicates with both PTP domains in the different IP versions.

The PTP system MAY include switches and routers. These devices MAY be Transparent Clocks, Boundary Clocks, or neither, in any combination. PTP Clocks MAY be Preferred timeTransmitters, Ordinary Clocks, or Boundary Clocks. The Ordinary Clocks may be TimeReceiver Only Clocks, or be timeTransmitter capable.

Note that PTP Ports will need to keep track of the Clock ID of received messages and not just the IP or Layer 2 addresses in any network that includes Transparent Clocks, or might include them in the future. This is important since Transparent Clocks might treat PTP messages that are altered at the PTP application layer as new IP

packets and new Layer 2 frames when the PTP messages are retransmitted. In IPv4 networks some clocks might be hidden behind a NAT, which hides their IP addresses from the rest of the network. Note also that the use of NATs may place limitations on the topology of PTP networks, depending on the port forwarding scheme employed. Details of implementing PTP with NATs are out of scope of this document.

PTP, similar to NTP, assumes that the one-way network delay for Sync messages and Delay Response messages are the same. When this is not true it can cause errors in the transfer of time from the timeTransmitter to the timeReceiver. It is up to the system integrator to design the network so that such effects do not prevent the PTP system from meeting the timing requirements. The details of network asymmetry are outside the scope of this document. See for example, ITU-T G.8271 [G8271].

## 6. Time Transfer and Delay Measurement

TimeTransmitter Clocks, Transparent Clocks and Boundary Clocks MAY be either one-step clocks or two-step clocks. TimeReceiver Clocks MUST support both behaviors. The End-to-End Delay measurement method MUST be used.

Note that, in IP networks, Sync messages and Delay Request messages exchanged between a timeTransmitter and timeReceiver do not necessarily traverse the same physical path. Thus, wherever possible, the network SHOULD be engineered so that the forward and reverse routes traverse the same physical path. Traffic engineering techniques for path consistency are out of scope of this document.

Sync messages MUST be sent as PTP event multicast messages (UDP port 319) to the PTP primary IP address. Two step clocks MUST send Follow-up messages as PTP general multicast messages (UDP port 320). Announce messages MUST be sent as multicast messages (UDP port 320) to the PTP primary address. The PTP primary IP address is 224.0.1.129 for IPv4 and FF0X:0:0:0:0:0:181 for IPv6, where X can be a value between 0x0 and 0xF. The different IPv6 address options are explained in IEEE 1588 [IEEE1588] Annex D, Section D.3. These addresses are allocated by IANA, see the Ipv6 Multicast Address Space Registry [IPv6Registry]

Delay Request messages MAY be sent as either multicast or unicast PTP event messages. TimeTransmitter Clocks MUST respond to multicast Delay Request messages with multicast Delay Response PTP general messages. TimeTransmitter Clocks MUST respond to unicast Delay Request PTP event messages with unicast Delay Response PTP general messages. This allows for the use of Ordinary Clocks which do not support the Enterprise Profile, if they are timeReceiver Only Clocks.

Clocks SHOULD include support for multiple domains. The purpose is to support multiple simultaneous timeTransmitters for redundancy. Leaf devices (non-forwarding devices) can use timing information from multiple timeTransmitters by combining information from multiple instantiations of a PTP stack, each operating in a different PTP Domain. Redundant sources of timing can be ensembled, and/or compared to check for faulty timeTransmitter Clocks. The use of multiple simultaneous timeTransmitters will help mitigate faulty timeTransmitters reporting as healthy, network delay asymmetry, and security problems. Security problems include on-path attacks such as delay attacks, packet interception / manipulation attacks. Assuming the path to each timeTransmitter is different, failures malicious or otherwise would have to happen at more than one path simultaneously. Whenever feasible, the underlying network transport technology SHOULD be configured so that timing messages in different domains traverse different network paths.

#### 7. Default Message Rates

The Sync, Announce, and Delay Request default message rates MUST each be once per second. The Sync and Delay Request message rates MAY be set to other values, but not less than once every 128 seconds, and not more than 128 messages per second. The Announce message rate MUST NOT be changed from the default value. The Announce Receipt Timeout Interval MUST be three Announce Intervals for Preferred TimeTransmitters, and four Announce Intervals for all other timeTransmitters.

The logMessageInterval carried in the unicast Delay Response message MAY be set to correspond to the timeTransmitter ports preferred message period, rather than 7F, which indicates message periods are to be negotiated. Note that negotiated message periods are not allowed, see forbidden PTP options (Section 13).

#### 8. Requirements for TimeTransmitter Clocks

TimeTransmitter Clocks MUST obey the standard Best TimeTransmitter Clock Algorithm from IEEE 1588 [IEEE1588]. PTP systems using this PTP Profile MAY support multiple simultaneous Grandmasters if each active Grandmaster is operating in a different PTP domain.



A PTP Port of a clock MUST NOT be in the timeTransmitter state unless the clock has a current value for the number of UTC leap seconds.

If a unicast negotiation signaling message is received it MUST be ignored.

In PTP Networks that contain Transparent Clocks, timeTransmitters might receive Delay Request messages that no longer contains the IP Addresses of the timeReceivers. This is because Transparent Clocks might replace the IP address of Delay Requests with their own IP address after updating the Correction Fields. For this deployment scenario timeTransmitters will need to have configured tables of timeReceivers' IP addresses and associated Clock Identities in order to send Delay Responses to the correct PTP Nodes.

#### 9. Requirements for TimeReceiver Clocks

In a network which contains multiple timeTransmitters in multiple domains, TimeReceivers SHOULD make use of information from all the timeTransmitters in their clock control subsystems. TimeReceiver Clocks MUST be able to function in such networks even if they use time from only one of the domains. TimeReceiver Clocks MUST be able to operate properly in the presence of a rogue timeTransmitter. TimeReceivers SHOULD NOT Synchronize to a timeTransmitter which is not the Best TimeTransmitter in its domain. TimeReceivers will continue to recognize a Best TimeTransmitter for the duration of the Announce Time Out Interval. TimeReceivers MAY use an Acceptable TimeTransmitter Table. If a timeTransmitter is not an Acceptable timeTransmitter, then the timeReceiver MUST NOT synchronize to it. Note that IEEE 1588-2019 requires timeReceiver Clocks to support both two-step or one-step timeTransmitter Clocks. See IEEE 1588 [IEEE1588], subClause 11.2.

Since Announce messages are sent as multicast messages timeReceivers can obtain the IP addresses of a timeTransmitter from the Announce messages. Note that the IP source addresses of Sync and Follow-up messages might have been replaced by the source addresses of a Transparent Clock, so, timeReceivers MUST send Delay Request messages to the IP address in the Announce message. Sync and Follow-up messages can be correlated with the Announce message using the Clock ID, which is never altered by Transparent Clocks in this PTP Profile.

#### 10. Requirements for Transparent Clocks

Transparent Clocks MUST NOT change the transmission mode of an Enterprise Profile PTP message. For example, a Transparent Clock MUST NOT change a unicast message to a multicast message. Transparent Clocks which synchronize to the timeTransmitter Clock might need to maintain separate clock rate offsets for each of the supported domains.

#### 11. Requirements for Boundary Clocks

Boundary Clocks SHOULD support multiple simultaneous PTP domains. This will require them to maintain separate clocks for each of the domains supported, at least in software. Boundary Clocks MUST NOT combine timing information from different domains.

#### 12. Management and Signaling Messages

PTP Management messages MAY be used. Management messages intended for a specific clock, i.e. the IEEE 1588 [IEEE1588] defined attribute targetPortIdentity.clockIdentity is not set to All 1s, MUST be sent as a unicast message. Similarly, if any signaling messages are used they MUST also be sent as unicast messages whenever the message is intended solely for a specific PTP Node.

#### 13. Forbidden PTP Options

Clocks operating in the Enterprise Profile MUST NOT use: Peer-to-Peer timing for delay measurement, Grandmaster Clusters, The Alternate TimeTransmitter option, Alternate Timescales. Unicast discovery, or unicast negotiation. Clocks operating in the Enterprise Profile MUST avoid any optional feature that requires Announce messages to be altered by Transparent Clocks, as this would require the Transparent Clock to change the source address and prevent the timeReceiver nodes from discovering the protocol address of the timeTransmitter.

#### 14. Interoperation with IEEE 1588 Default Profile

Clocks operating in the Enterprise Profile will interoperate with clocks operating in the Default Profile described in IEEE 1588 [IEEE1588] Annex I.3. This variant of the Default Profile uses the End-to-End delay measurement mechanism. In addition, the Default Profile would have to operate over IPv4 or IPv6 networks, and use management messages in unicast when those messages are directed at a specific clock. If either of these requirements are not met than Enterprise Profile clocks will not interoperate with Annex I.3 Default Profile Clocks. The Enterprise Profile will not interoperate with the Annex I.4 variant of the Default Profile which requires use

of the Peer-to-Peer delay measurement mechanism.

Enterprise Profile Clocks will interoperate with clocks operating in other PTP Profiles if the clocks in the other PTP Profiles obey the rules of the Enterprise Profile. These rules MUST NOT be changed to achieve interoperability with other PTP Profiles.

## 15. Profile Identification

The IEEE 1588 standard requires that all PTP Profiles provide the following identifying information.

```
PTP Profile:
Enterprise Profile
Profile number: 1
Version: 1.0
Profile identifier: 00-00-5E-01-01-00
```

This PTP Profile was specified by the IETF

A copy may be obtained at  
<https://datatracker.ietf.org/wg/tictoc/documents>

## 16. Acknowledgements

The authors would like to thank Richard Cochran, Kevin Gross, John Fletcher, Laurent Montini and many other members of IETF for reviewing and providing feedback on this draft.

This document was initially prepared using 2-Word-v2.0.template.dot and has later been converted manually into xml format using an xml2rfc template.

## 17. IANA Considerations

There are no IANA requirements in this specification.

## 18. Security Considerations

Protocols used to transfer time, such as PTP and NTP can be important to security mechanisms which use time windows for keys and authorization. Passing time through the networks poses a security risk since time can potentially be manipulated. The use of multiple simultaneous timeTransmitters, using multiple PTP domains can mitigate problems from rogue timeTransmitters and on-path attacks. Note that Transparent Clocks alter PTP content on-path, but in a manner specified in IEEE 1588-2019 [IEEE1588] that helps with time transfer accuracy. See sections 9 and 10. Additional security

mechanisms are outside the scope of this document.

PTP native management messages SHOULD NOT be used, due to the lack of a security mechanism for this option. Secure management can be obtained using standard management mechanisms which include security, for example NETCONF [RFC6241].

General security considerations of time protocols are discussed in RFC 7384 [RFC7384].

## 19. References

### 19.1. Normative References

- [IEEE1588] Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2019, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems.", November 2019, <<https://www.ieee.org>>.
- [IEEE1588g] Institute of Electrical and Electronics Engineers, "IEEE std. 1588g-2022, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Amendment 2: Master-Slave Optional Alternative Terminology", December 2022, <<https://www.ieee.org>>.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 2119, DOI 10.17487/RFC2119, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

## 19.2. Informative References

- [Estrela\_and\_Bonebakker]  
Estrela, P. and L. Bonebakker, "Estrela and Bonebakker, "Challenges deploying PTPv2 in a global financial company"", DOI 10.1109/ISPCS.2012.6336634, 2012, <[https://www.researchgate.net/publication/260742322\\_Challenges\\_deploying\\_PTPv2\\_in\\_a\\_global\\_financial\\_company](https://www.researchgate.net/publication/260742322_Challenges_deploying_PTPv2_in_a_global_financial_company)>.
- [G8271] International Telecommunication Union, "ITU-T G.8271/Y.1366, "Time and Phase Synchronization Aspects of Packet Networks"", March 2020, <<https://www.itu.int>>.
- [IPv6Registry]  
Venaas, S., "IPv6 Multicast Address Space Registry", February 2024, <<https://iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>>.
- [ISPCS] Arnold, D., "Plugfest Report", October 2017, <<https://www.ispcs.org>>.
- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, DOI 10.17487/RFC2026, October 1996, <<https://www.rfc-editor.org/info/rfc2026>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.

## Authors' Addresses

Doug Arnold  
Meinberg-USA  
3 Concord Rd  
Shrewsbury, Massachusetts 01545  
United States of America  
Email: [doug.arnold@meinberg-usa.com](mailto:doug.arnold@meinberg-usa.com)

Heiko Gerstung  
Meinberg  
Lange Wand 9  
31812 Bad Pyrmont  
Germany  
Email: [heiko.gerstung@meinberg.de](mailto:heiko.gerstung@meinberg.de)

Network Time Protocol  
Internet-Draft  
Intended status: Standards Track  
Expires: 21 August 2024

M. Langer  
PTB  
R. Bermbach  
Ostfalia University  
18 February 2024

NTS4PTP – Key Management System for the Precision Time Protocol Based on  
the Network Time Security Protocol  
draft-langer-ntp-nts-for-ntp-07

#### Abstract

This document specifies an automatic key management service for the integrated security mechanism (prong A) of IEEE Std 1588-2019 (PTPv2.1) described there in Annex P. This key management follows the immediate security processing approach of prong A and extends the NTS Key Establishment protocol defined in IETF RFC 8915 for securing NTPv4. The resulting NTS for PTP (NTS4PTP) protocol provides a security solution for all relevant PTP modes and operates completely independent of NTPv4.

#### Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 August 2024.

#### Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1.	Introduction . . . . .	3
1.1.	Requirements Language . . . . .	7
1.2.	Terms and Abbreviations . . . . .	7
2.	Key Management for PTP Using Network Time Security . . . . .	10
2.1.	Setup of a TLS Communication Channel with the NTS-KE Protocol . . . . .	11
2.2.	Setup of a TLS Communication Channel with the NTS-TSR Protocol . . . . .	12
2.3.	NTS Message Exchange for Group-Based Mode . . . . .	13
2.4.	NTS Message Exchange for the Ticket-Based Mode . . . . .	15
2.5.	General Topics . . . . .	19
2.5.1.	Key Update Process . . . . .	19
2.5.2.	Key Generation . . . . .	24
2.5.3.	Time Information of the NTS-KE server . . . . .	24
2.5.4.	Certificates . . . . .	24
2.5.5.	Upfront Configuration . . . . .	25
2.5.5.1.	Security Parameters . . . . .	25
2.5.5.2.	Key Lifetimes . . . . .	26
2.5.5.3.	Certificates . . . . .	26
2.5.5.4.	Authorization . . . . .	26
2.5.5.5.	Transparent Clocks . . . . .	27
2.5.5.6.	Start-up considerations . . . . .	28
3.	NTS Messages for PTP . . . . .	28
3.1.	PTP Key Request Message . . . . .	29
3.2.	PTP Key Response Message . . . . .	30
3.3.	PTP Registration Request Message . . . . .	32
3.4.	PTP Registration Response Message . . . . .	33
3.5.	PTP Registration Revoke Message . . . . .	35
3.6.	PTP Heartbeat Message . . . . .	36
4.	NTS Records for PTP . . . . .	37
4.1.	Overview of the NTS Records . . . . .	39
4.2.	Detailed Description of the NTS Records . . . . .	41
4.2.1.	AEAD Algorithm Negotiation . . . . .	41
4.2.2.	Association Mode . . . . .	42
4.2.3.	Current Parameters . . . . .	46
4.2.4.	End of Message . . . . .	48
4.2.5.	Error . . . . .	48
4.2.6.	Heartbeat Timeout . . . . .	49
4.2.7.	Next Parameters . . . . .	50
4.2.8.	NTS Next Protocol Negotiation . . . . .	51
4.2.9.	NTS Message Type . . . . .	52



4.2.10. PTP Time Server . . . . .	53
4.2.11. Security Association . . . . .	55
4.2.12. Source PortIdentity . . . . .	56
4.2.13. Status . . . . .	57
4.2.14. Supported MAC Algorithms . . . . .	58
4.2.15. Ticket . . . . .	60
4.2.16. Ticket Key . . . . .	62
4.2.17. Ticket Key ID . . . . .	63
4.2.18. Validity Period . . . . .	63
5. Additional Mechanisms . . . . .	65
5.1. Replay Protection . . . . .	66
5.2. AEAD Operation . . . . .	67
5.3. SA/SPP Management . . . . .	69
6. New TICKET TLV for PTP Messages . . . . .	70
7. AUTHENTICATION TLV Parameters . . . . .	72
8. IANA Considerations . . . . .	73
9. Security Considerations . . . . .	74
10. Acknowledgements . . . . .	74
11. References . . . . .	74
11.1. Normative References . . . . .	74
11.2. Informative References . . . . .	75
Authors' Addresses . . . . .	76

## 1. Introduction

In its Annex P the IEEE Std 1588-2019 ([IEEE1588-2019], Precision Time Protocol version 2.1, PTPv2.1) defines a comprehensive PTP security concept based on four prongs (A to D). Prong A incorporates an immediate security processing approach and specifies in section 16.14 an extension to secure PTP messages by means of an AUTHENTICATION TLV (AuthTLV) containing an Integrity Check Value (ICV). For PTP instances to use the securing mechanism, a respective key needs to be securely distributed among them. Annex P gives requirements for such a key management system and mentions potential candidates without further specification, but allows other solutions as long as they fulfill those requirements.

Since many time server appliances support both, the Precision Time Protocol (PTP) and the Network Time Protocol (NTP), it should be easier for the manufacturer of these devices and the network operator if PTP and NTP use a key management system based on the same technology. The Network Time Security (NTS) protocol was specified by the Internet Engineering Task Force (IETF) to protect the integrity of NTP messages [RFC8915]. Its NTS Key Establishment sub-protocol is secured by the Transport Layer Security (TLS 1.3, IETF RFC 8446 [RFC8446]) mechanism. TLS is used to protect numerous popular network protocols, so it is present in many networks.

This document specifies an automatic key management service, NTS for PTP, short NTS4PTP, for the immediate security processing in prong A. The solution [Langer\_et\_al.\_2022], [Langer\_et\_al.\_2020] is based on and expands the NTS Key Establishment protocol defined in IETF RFC 8915 [RFC8915] for securing NTP, but works completely independent of NTP. In addition, this document introduces a new sub-protocol, the NTS Time Server Registration (NTS-TSR) protocol, defining the communication between PTP unicast servers (grantors) with the NTS-Key Establishment server (NTS-KE server). (In NTS for NTP the specification of the communication between NTS time server and NTS-KE server has been left open.) Figure 1 depicts the participants of the NTS4PTP protocol and the sub-protocols they use.

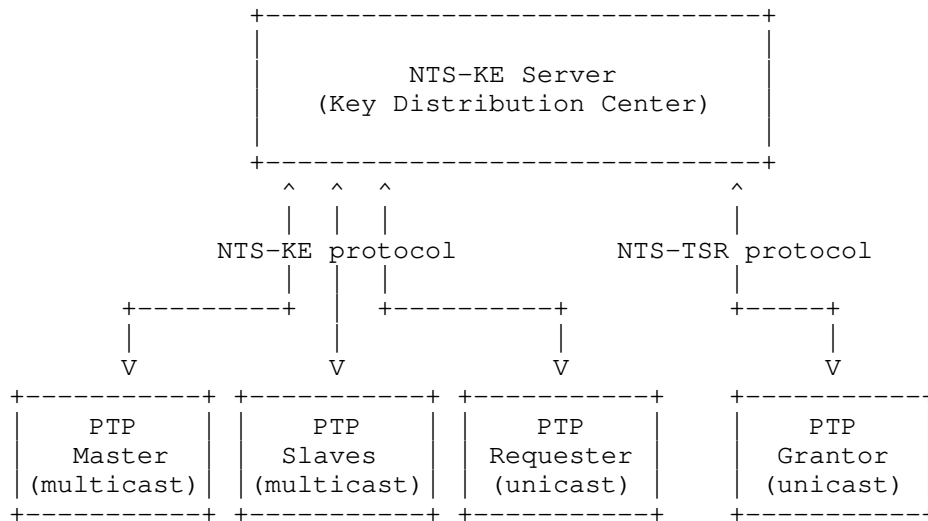


Figure 1: Communication of PTP instances with the NTS-KE server using the NTS-KE and NTS-TSR sub-protocols

For PTP multicast communication the PTP grandmaster as well as all participating PTP slaves use the NTS-KE protocol to obtain the security association (SA), i.e. key, lifetime etc. for a specific group. PTPv2.1 does not know groups, but distinguishes between PTP domains and profiles in order to separate different PTP networks from each other. NTS4PTP derives groups from this information to assign the logically separated PTP networks to their own SA (see first paragraph of Section 2.3). For such PTP multicast or mixed multicast/unicast communication, NTS4PTP defines the group-based mode, short GrM.

For securing a PTP unicast communication a potential grantor (time server) uses the NTS-TSR protocol to register with the NTS-KE server. Thereby, ticket key, lifetime etc. for encrypting a so-called ticket are exchanged. A potential PTP unicast client (requester) then again uses the NTS-KE protocol to obtain the security association, i.e. unicast key, lifetime etc., as well as an encrypted ticket for the unicast communication with the specific grantor from the NTS-KE server. Thereafter, the ticket is transported from requester to grantor attached to a PTP signaling message [IEEE1588-2019] to establish a so-called unicast contract for delivering PTP time information. The (ticket key-) encrypted ticket holds all necessary information for the grantor to identify the requester as well as the (unicast) key used to secure and check the PTP messages between them. For this PTP unicast communication (also called negotiated PTP unicast), NTS4PTP defines the ticket-based mode, short TiM.

Though the key management for PTP is based on the NTS Key Establishment (NTS-KE) protocol for NTP, it works completely independent of NTP. The key management system uses the procedures described in IETF RFC 8915 for the NTS-KE protocol and expands it with new NTS messages for PTP. It may be applied in a key establishment server that already manages NTP but can also be operated only handling key establishment for PTP. Even when the PTP network is isolated from the Internet, a key establishment server can be installed in that network providing the PTP instances with necessary key and security parameters.

The NTS-KE server may often be implemented as a separate unit. It also may be collocated with a PTP instance, e.g., the Grandmaster. In the latter case communication between the NTS-KE server program and the PTP instance program needs to be implemented in a secure way if TLS communication (e.g., via local host or inter-process communication) is not or cannot be used.

Using the expanded NTS Key Establishment protocol and the newly defined NTS Time Server Registration protocol for the NTS key management for PTP, NTS4PTP provides the two principle approaches specified in this document:

#### 1. Group-based mode (GrM)

- \* very suitable for PTP multicast mode and mixed multicast/unicast mode,
- \* definition of one or more security groups in the PTP network,
- \* secured 1:n communication
- \* suitable for unicast mode in small subgroups of very few participants (Group-of-2, Go2, see Section 2.3) but poor scaling and more administration work,



### 1.1. Requirements Language

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **NOT RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 1.2. Terms and Abbreviations

Term	Description
AEAD	Authenticated Encryption with Associated Data [RFC5116]
AES	Advanced Encryption Standard, also: Rijndael
Authentication TLV (AuthTLV)	PTPv2.1 extension that provides authenticity and integrity protection for PTP messages [IEEE1588-2019]
ALPN	Application-Layer Protocol Negotiation [RFC7301]
CMAC	Cipher-based Message Authentication Code, see also MAC
Container, Container records	Container records (short: container) comprise a set of NTS records in its record body that serve a specific purpose, e.g., the Current Parameters container record.
CSPRNG	Cryptographically Secure Pseudorandom Number Generator
DoS	Denial of Service
DDoS	Distributed Denial of Service
GMAC	Galois Message Authentication Code, see also MAC
Go2	Group-of-2, group-based unicast mode for small subgroups of very few participants
GP, Grace Period	Defines a period of time during which security parameters are accepted for a short time after

	their lifetime has expired
Group	NTS4PTP uses the term to describe PTP entities in a PTP multicast setup (e.g., master, slave, ...) that are authorized for a common security association to secure and verify PTP messages between them.
GrM	Group-based mode of NTS4PTP
Group Key	Key used for authentication of PTP messages in group-based mode (GrM)
HMAC	Hash-based Message Authentication Code, see also MAC
ICV	Integrity Check Value, result of a cryptographic function used to detect unauthorized modifications of a PTP message, field in the Authentication TLV
IEEE 802.3	Standards collection defining the physical layer and data link layer's media access control (MAC) of wired Ethernet, transport mode in PTP
IP, IPv4, IPv6	Internet Protocol, network layer communications protocol, version 4 or version 6, part of the Internet protocol suite
IV	Initialization Vector, for example used with some MAC algorithms
Lifetime	Specifies the validity period of the security parameters in seconds, which is counted down
MAC address	Medium Access Control address, unique identifier used as a network address within a network segment
MAC algorithm	Message Authentication Code, short piece of information used for authenticating and integrity-checking of a message
NTP	Network Time Protocol [RFC5905]
NTS4PTP	NTS for PTP, variant of NTS to provide key management to PTP

NTS	Network Time Security [RFC8915]
NTS-KE	Network Time Security Key Establishment protocol
NTS-TSR	Network Time Security Time Server Registration protocol
OCSP	Online Certificate Status Protocol [RFC6960]
PKI	Public Key Infrastructure
PortIdentity	Specifies a specific PTP port
PTP	Precision Time Protocol [IEEE1588-2019]
Record, NTS record	Special NTS type-length-value data structure defining specific parameters; records build the respective NTS messages (differs from the TLV format of PTP)
SA	Security Association, description of the set of security parameters necessary to provide security services (e.g., authentication and integrity) between different entities sharing the same SA
SAD	Security Association Database
sdoId	Standards Development Organization Identifier, attribute for providing isolation of PTP Instances using different PTP profiles; in NTS4PTP it forms the group number in combination with the PTP domain number
SPP	Security Parameter Pointer
TCP	Transmission Control Protocol, part of the Internet protocol suite
Ticket	NTS record which contains the encrypted security parameters that a grantor needs for a secured PTP unicast connection to the requester
Ticket Key	Encryption key for the ticket, negotiated between NTS-KE server and grantor during the

	registration process (different from unicast key)
TC, Transparent Clock	Device in a PTP network with multiple PTP ports (switch) which measures its transit time and provides it in a correction field of the PTP message
Ticket TLV	TLV for carrying the ticket from requester to grantor within a PTP Signaling message for unicast request
TiM	Ticket-based mode for NTS4PTP
TLS	Transport Layer Security [RFC8446]
TLV	Data set containing a type, length, and value field. Used in PTPv2.1 [IEEE1588-2019], compare to Authentication TLV and Ticket TLV
UDP	User Datagram Protocol, part of the Internet protocol suite
Unicast Key	Used to secure the PTP messages between requester and grantor (different from ticket key)
UP, Update Period	During the update period new security parameters are available at the NTS-KE server, resp. grantors should re-register with the NTS-KE server
X.509	Standard to form X.509 certificates [ITU-T_X.509]

Table 1: Terms and abbreviations

## 2. Key Management for PTP Using Network Time Security

After the rundown of the different PTP instances and the sub-protocols they use for communication with the NTS Key Establishment (NTS-KE) server in the introduction, the following sections specify the setup and use of TLS 1.3 to secure the communication with the NTS-KE server, before the message exchange for both, the group-based mode as well as the ticket-based mode is described in detail in Section 2.3 and Section 2.4. More general topics as key update, authentication and authorization etc. are covered in Section 2.5.



## 2.1. Setup of a TLS Communication Channel with the NTS-KE Protocol

TLS is a layer five protocol that runs on TCP over IP. Therefore, PTP implementations that support NTS-based key management need to support TCP and IP (at least on a separate management port).

A PTP instance wanting to request a key using the NTS-KE protocol defined in [RFC8915], first starts a TLS 1.3 connection to the NTS-KE server.

The PTP instance connects to the NTS-KE server on the NTS TCP port (port number 4460). Then both parties perform a TLS handshake to establish a TLS 1.3 communication channel. The details of the TLS handshake are specified in IETF RFC 8446 [RFC8446].

Implementations MUST conform to the rules stated in Section 3 TLS Profile for Network Time Security of IETF RFC 8915 [RFC8915]:

```
_"Network Time Security makes use of TLS for NTS key
establishment._
_Since the NTS protocol is new as of this publication, no
backward-compatibility concerns exist to justify using obsolete,
insecure, or otherwise broken TLS features or versions._
_Implementations MUST conform with RFC 7525_ [RFC7525]_or with a
later revision of BCP 195._
_Implementations MUST NOT negotiate TLS versions earlier than
1.3_[RFC8446]_and MAY refuse to negotiate any TLS version that has
been superseded by a later supported version._
_Use of the Application-Layer Protocol Negotiation
Extension_[RFC7301]_is integral to NTS, and support for it is
REQUIRED for interoperability ... "_
```

The client starts the TLS 1.3 handshake with a 'Client Hello' message to the NTS-KE server containing the Application Layer Protocol Negotiation (ALPN) [RFC7301] extension containing "ntske/1", which refers to the NTS Key Establishment as the subsequent protocol. The server responds with a "Server Hello" message sending its certificate and feasible cipher suites as well as requesting the client's certificate using a TLS 'CertificateRequest'. (The latter does not conflict to the procedure in NTS for NTP.)

Afterwards, the client authenticates the server using the root CA certificate or by means of the Online Certificate Status Protocol (OCSP, IETF RFC 6960) [RFC6960]. In the same way, the server authenticates the client, if it had sent its certificate (which is always necessary with NTS4PTP, in contrast to NTS for NTP.) After the authentication procedure both, client and server agree on the cipher suite and then establish a secured channel that ensures authenticity, integrity and confidentiality for subsequent NTS messages.

Once the TLS session is established, the PTP instance will ask for a group key as well as the associated security parameters using the new NTS message PTP Key Request (see Section 3.1). The NTS-KE server will respond with a PTP Key Response message (see Section 3.2).

The NTS-KE server keeps the TLS session open until a short timeout configured by the admin expires or the 'close notify' arrives. This allows the PTP instance to make another NTS request without starting a new TLS handshake. Finally, the NTS-KE server also sends a 'close notify' to the PTP instance and closes the TLS channel.

With the key and other information received, the PTP instance can take part in the secured PTP communication in the different modes of operation.

After the reception of the first set of security parameters the PTP instance may resume the TLS session according to IETF RFC 8446 [RFC8446], Section 4.6.1, allowing the PTP instance to skip the TLS version and algorithm negotiations. If TLS Session Resumption ([RFC8446], Section 2.2) is used and supported by the NTS-KE server, a suitable lifetime (max. 24 hrs) for the TLS session key must be defined to not open the TLS connection for security threats. If the NTS-KE server does not support TLS resumption, a full TLS handshake must be performed.

As the TLS session provides authentication, but not authorization additional means have to be used for the latter (see Section 2.5.5.4).

## 2.2. Setup of a TLS Communication Channel with the NTS-TSR Protocol

As already mentioned and shown in Figure 1 and Figure 2, the new NTS Time Server Registration protocol is used for registering a grantor with the NTS-KE server (ticket-based mode only). Thereby, the new messages PTP Registration Request message, PTP Registration Response message, PTP Registration Revoke message and PTP Heartbeat message are applied.

The setup of the TLS channel in this ticket-based mode (TiM) is handled in the same way as described above for the NTS-KE protocol, see Section 2.1. The only difference lies in the ALPN used, which is now "ntstsr/1".

Once the TLS session is established, the grantor will register with the NTS-KE server using the NTS message PTP Registration Request (see Section 3.3). The NTS-KE server will respond with a PTP Registration Response message (see Section 3.4) containing ticket key, lifetime etc.

When the PTP Registration Request message was responded with a PTP Registration Response, the TLS session can be closed with a 'close notify' TLS alert from the grantor followed by a 'close notify' of the NTS-KE server.

Also using a TLS connection with the NTS-KE server a grantor can cancel its registration with a PTP Registration Revoke message (see Section 3.5). With a PTP Heartbeat message (see Section 3.6) the grantor signals continuously its availability to the NTS-KE server.

### 2.3. NTS Message Exchange for Group-Based Mode

As described in Section 2.1, a PTP instance wanting to join a secured PTP communication in the group-based modes contacts the NTS-KE server starting the establishment of a secured TLS connection using the NTS-KE protocol (ALPN: ntske/1). Then, the client continues with a PTP Key Request message (see Section 3.1), asking for a specific group as shown in Figure 3. The NTS-KE server identifies the respective sub-protocol by means of the ALPN and analyses the contents of the Next Protocol Negotiation record. If it is PTP the server examines whether the client had sent its certificate and that it is valid. Finally, it checks the authorization of the client. If everything is ok the NTS-KE server generates the respective PTP Key Response message (see Section 3.2) for the requesting client with all the necessary data to join the group communication. Else, it contains a respective error code if the PTP instance is not allowed to join the group. This procedure is necessary for all parties, which are or will be members of that PTP group including the Grandmaster and other special participants, e.g., Transparent Clocks. As mentioned above, this not only applies to multicast mode but also to mixed multicast/unicast mode (former hybrid mode) where the explicit unicast communication uses the multicast group key received from the NTS-KE server. The group number for both modes is primarily generated by a concatenation of the PTP domain number and the PTP profile identifier (sdoId - Standards Development Organization Identifier), as described in Section 4.2.2.

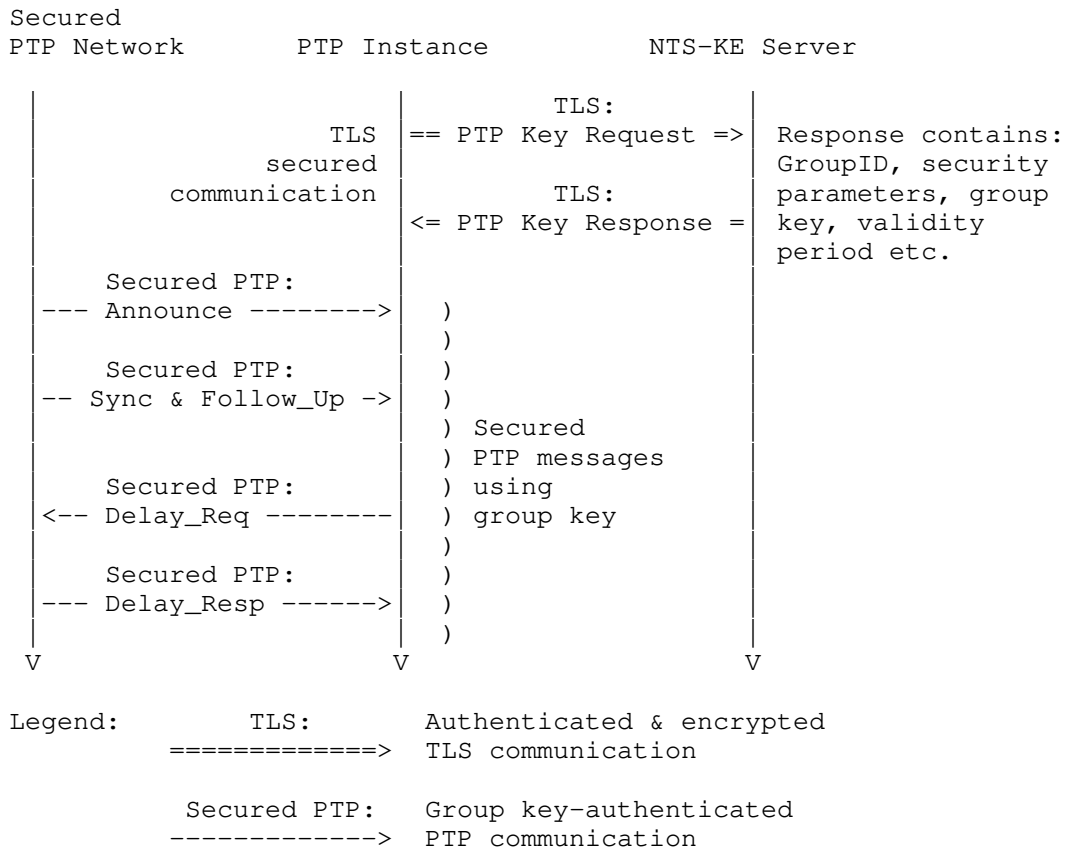


Figure 3: Message exchange for the group-based mode (GrM)

Additionally, besides multicast and mixed multicast/unicast mode, a group of two (or few more) PTP instances can be configured, practically implementing a special group-based unicast communication mode, the group-of-2 (Go2) mode.

This Go2 mode requires additional administration in advance defining groups-of-2 and supplying them with an additional attribute in addition to the group number mentioned for the other group-based modes the subGroup attribute in the Association Mode record (see Section 4.2.2) of the PTP Key Request message. So, addressing for Go2 is achieved by use of the group number derived from domain number, sdoId, and the additional attribute subGroup. Communication in that mode is performed using multicast addresses. If the latter is undesirable, unicast addresses can be used but the particular IP or MAC addresses of the communication partners need to be configured upfront, too.

In spite of its specific name, Go2 allows more than two participants, for example additional Transparent Clocks. All participants in that subgroup need to be configured respectively. (To enable the NTS-KE server to supply the subgroup members with the particular security data the respective certificates may reflect permission to take part in the subgroup. Else another authorization method is to be used.)

Having predefined the Go2s the key management for this mode of operation follows the same procedure (see Figure 3) and uses the same NTS messages as the other group-based modes. Both participants, the Group-of-2 requester and the respective grantor need to have received their security parameters including key etc. before secure PTP communication can take place.

After the NTS Key Establishment messages for these group-based modes have been exchanged, the secured PTP communication can take place using the security association(s) communicated. The participants of the PTP network are now able to use the group key to verify secured PTP messages of the corresponding group or to generate secured PTP messages itself. In order to do this, the PTP node applies the group key together with the MAC algorithm to the PTP packet to generate the ICV transported in the AUTHENTICATION TLV of the PTP message.

The key management for these modes works relatively simple and needs only the above mentioned two NTS messages: PTP Key Request and PTP Key Response.

#### 2.4. NTS Message Exchange for the Ticket-Based Mode

The scaling problems of the group-based mode are solved by the ticket-based mode (TiM) for unicast connections. TiM ensures end-to-end security between the two PTP communication partners, requester and grantor, and is therefore only suitable for PTP unicast where no group binding exists. Thus, this model scales excellently with the number of connections. TiM also allows free MAC algorithm and server negotiation, eliminating the need for the administrator to manually

prepare the table of acceptable unicast masters at each individual PTP node. In addition, this allows optional load control by the NTS-KE server.

In PTP unicast mode using unicast message negotiation ([IEEE1588-2019], Section 16.1) any potential instance (the grantor) which can be contacted by other PTP instances (the requesters) needs to register upfront with the NTS-KE server as depicted in Figure 4. For the registration, again a TLS channel has to be set up using the new NTS Time Server Registration sub-protocol with the ALPN "ntstsr/1" as described in Section 2.2. This also ensures the mutual authentication of grantor and NTS-KE server.

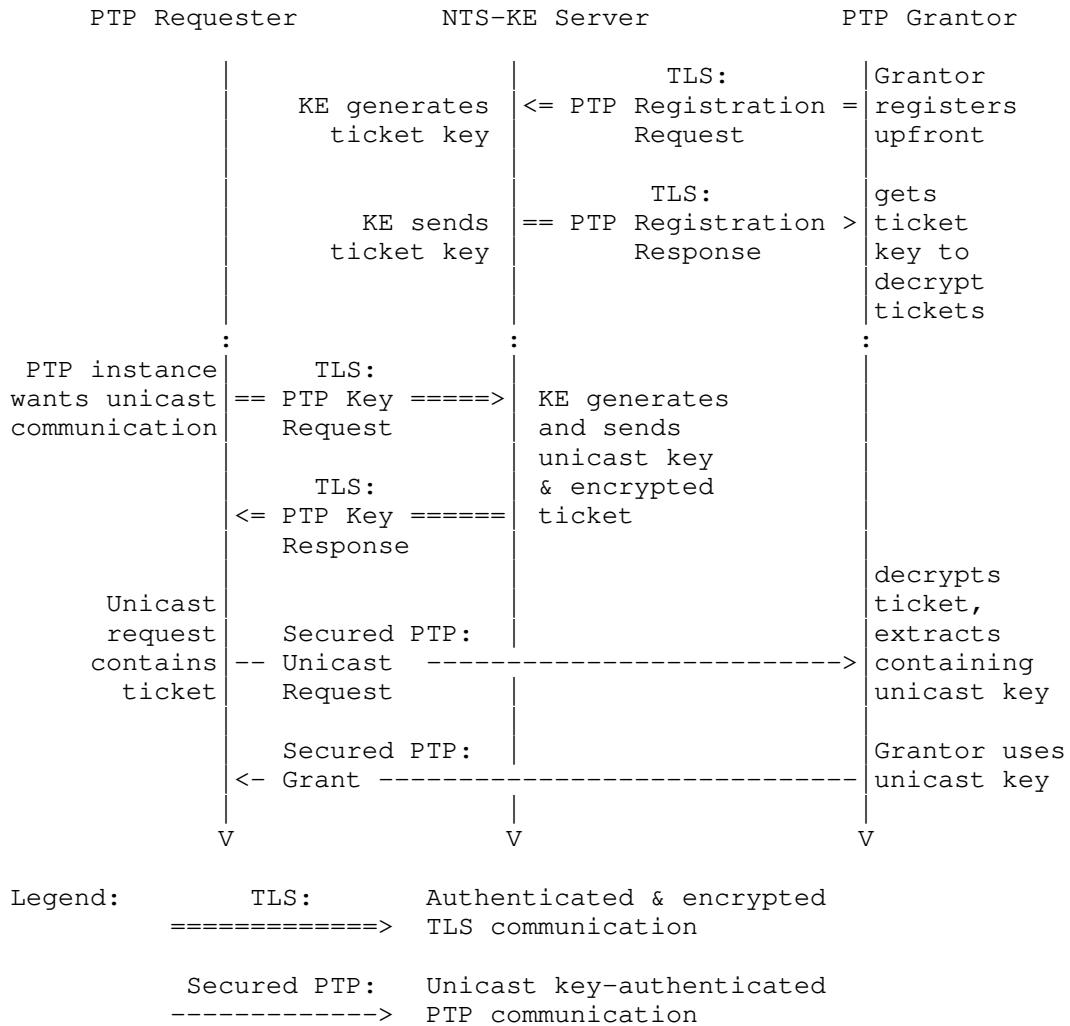


Figure 4: Message exchange for ticket-based unicast mode (TiM)

\_(Note: As any PTP instance may request unicast messages from any other instance the terms requester and grantor as used in the standard suit better than talking about slave respectively master. In unicast PTP, the grantor is typically a PTP Port in the MASTER state, and the requester is typically a PTP Port in the SLAVE state. However, all PTP Ports are allowed to grant and request unicast PTP message contracts regardless of which state they are in. A PTP port in MASTER state may be requester, a port in SLAVE state may be a grantor.)\_

The registration of a PTP grantor is performed via a PTP Registration Request message (see Section 3.3). The NTS-KE server answers with a PTP Registration Response message (see Section 3.4). If no delivery of security data is possible for whatever reason, the PTP Registration Response message contains a respective error code.

With the reception of the PTP Registration Response message, the grantor holds a ticket key known only to the NTS-KE server and the registered grantor. With this ticket key it can decrypt cryptographic information contained in a so-called ticket which enables secure unicast communication.

After the end of the registration process (phase 1), phase 2 begins with the key request of the client (here called requester). Similar to the group-based mode, a PTP instance (the requester) wanting to start a secured PTP unicast communication with a specific grantor contacts the NTS-KE server sending a PTP Key Request message (see Section 3.1) as shown in Table 2, again using the TLS-secured NTS Key Establishment protocol. The NTS-KE server performs the authentication check of the client and then answers with a PTP Key Response message (see Section 3.2) with all the necessary data to begin the unicast communication with the desired partner or with a respective error code if unicast communication with that instance is unavailable. Though the message types are the same as in GrM the content differs.

In TiM the PTP Key Response message includes a unicast key to secure the PTP message exchange with the desired grantor. In addition, it contains the above mentioned (partially) encrypted ticket which the requester later (phase 3) transmits in a special Ticket TLV (see Section 6) with the secured PTP message to the grantor.

After the NTS Key Establishment messages for the PTP unicast mode have been exchanged, finally, the secured PTP communication (phase 3) can take place using the security association(s) communicated. A requester may send a (unicast key-) secured PTP signaling message containing the received encrypted ticket, asking for a grant of a so-called unicast contract which contains a request for a specific PTP message type, as well as the desired frame rate.

The grantor receiving the PTP message decrypts the received ticket with its ticket key and extracts the containing security parameters, for example the unicast key used by the requester to secure the PTP message and the requesters identity. In that way the grantor can check the received message, identify the requester and can use the unicast key for further secure PTP communication with the requester until the unicast key expires.



A grantor that supports unicast and provides sufficient capacity will acknowledge the request for a unicast contract with a PTP unicast grant.

If a grantor is no longer at disposal for unicast mode during the lifetime of registration and ticket key, it sends a TLS-secured PTP Registration Revoke message (see Section 3.5, not shown in Figure 4) to the NTS-KE server, so requesters no longer receive security associations (key etc.) in PTP Key Response messages for this grantor. Either the NTS-KE server sends response messages with SAs of other grantors or with respective error codes.

The PTP Heartbeat message (see Section 3.6, not shown in Figure 4) allows grantors to send messages to the NTS-KE server at regular intervals during the validity of the current security data and to signal their own functionality. Optionally, these messages can contain status reports, for example, to enable load balancing between the registered time servers or to provide additional monitoring.

With its use of the two sub-protocols, the NTS-KE and the NTS-TSR protocol, this unicast mode (TiM) is a bit more complex than the Group-of-2 mode and eventually uses all six new NTS messages. However, no subgroups have to be defined upfront. Addressing a grantor, the requesting instance simply may use the grantor's IP, MAC address or PortIdentity attribute.

## 2.5. General Topics

This section describes more general topics like key update and key generation as well as discussion of the time information on the NTS-KE server, the use of certificates and topics concerning upfront configuration.

### 2.5.1. Key Update Process

The security parameters update process is an important part of NTS4PTP. It keeps the keys up to date, allows for both, runtime security policy changes and easy group control. The rotation operation allows uninterrupted PTP operation in multicast as well as unicast mode.

The update mechanism is based on the Validity Period record in the NTS response messages, which includes the three values lifetime, update period (UP) and grace period (GP), see Figure 5. The lifetime parameter specifies the validity period of the security parameters (e.g., security association (SA) and ticket) in seconds, which is counted down. This value can range from a few minutes to a few days. (Due to the design of the replay protection, a maximum lifetime of up



As the value for lifetime is specified in seconds which denote the remaining time and is decremented down to zero, hard adjustments of the clock used have to be avoided. Therefore, the use of a monotonic clock is recommended. Requests during the currently running validity period will receive respectively adapted count values.

The Validity Period record (see Section 4.2.18) with its parameters lifetime, update period and grace period is contained in a so-called Current Parameters container record. Together with other security parameters this container record is always present in a PTP Key respectively Registration Response message. During the update period the response message additionally comprises the Next Parameters container record, which holds the new lifetime etc. starting at the end of the current lifetime as well as the other security parameters of the upcoming lifetime cycle.

Any PTP client sending a PTP Key Request to the NTS-KE server, be it in GrM to receive the group SA or be it in TiM asking for the unicast SA (unicast key etc. and encrypted ticket), will receive the Current Parameters container record where lifetime includes the remaining time to run rather than the full. Requesting during the update period the response includes also the new lifetime value etc. in the Next Parameters container record. The new lifetime is the full value of the validity starting at the end of the current lifetime and update period. After the old lifetime has expired, only the new parameters (including lifetime, update period and grace period) have to be used. Merely during the grace period, the old SA will be accepted to cope with smaller delays in the PTP communication.

All PTP clients are obliged to connect to the NTS-KE server during the update period to allow for uninterrupted secured PTP operation. To avoid peak load on the NTS-KE server all clients SHOULD choose a random starting time during the update period.

In TiM the unicast grantors execute the NTS-TSR protocol to register with the NTS-KE server. The rotation sequence (see Figure 6) and the behavior of the PTP Registration Response message is almost identical to the NTS-KE protocol. The main difference here is that the update period has to start earlier so that a grantor has re-registered before requesters ask for new security parameters at the NTS-KE server.

As the difference between the start of the requesters update period and the beginning of the update period of the grantor is not communicated, the grantor should contact the NTS-KE server directly after the start of its update period. However, since the rotation periods occur at different times for multiple grantors, no load peaks occur here either.

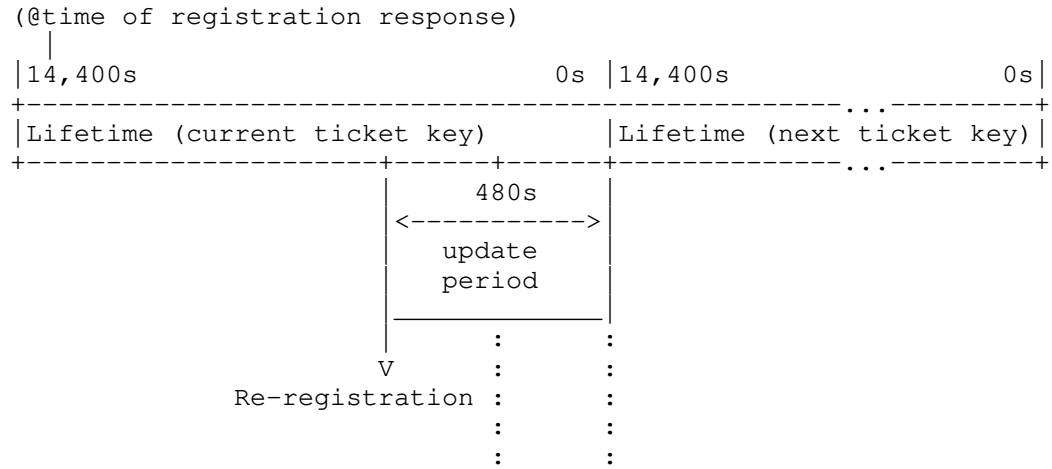
If a grantor does not re-register in time, requesters asking for a key etc. may not receive a Next Parameters container record, as no new SA is available at that point. So, requesters need to try again later.

As unicast contracts in TiM run independently of the update cycle, a special situation may occur. If the remaining lifetime is short, the grantor decides whether it grants any contract longer than the remaining lifetime or not. If a unicast contract is to be extended within the update period and the requester already owns the new ticket, it can already apply the upcoming security parameters here. This corresponds to some kind of negative grace period (pre-validity use of upcoming security parameters) and allows the requester to negotiate the full time for the unicast contract with the grantor.

If a grantor has revoked his registration with a PTP Registration Revoke message, requesters will receive a PTP Key Response message with an error code when trying to update for a new unicast key. No immediate key revoke mechanism exists. The grantor SHOULD NOT grant respective unicast requests during the remaining lifetime of the revoked key.

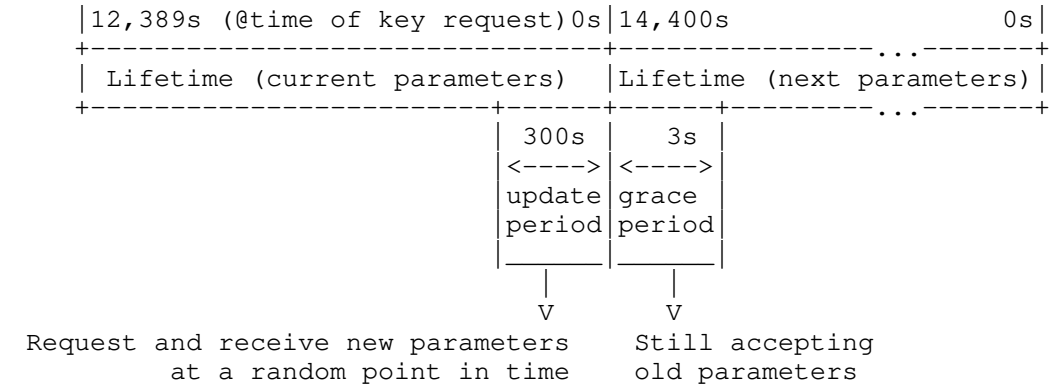
Update process grantor:

-----



Update process requester:

-----



Example:

-----

lifetime (full):           14,400s = 4h  
update period grantor:     480s = 8min  
update period requester:   300s = 5min  
grace period:                3s

Figure 6: Example of the parameter rotation using lifetime and update period in ticket-based mode

### 2.5.2. Key Generation

In all cases keys obtained by a secure random number generator SHALL be used. The length of the keys depends on the cryptographic algorithm used (see also last subsection in Section 5.3).

### 2.5.3. Time Information of the NTS-KE server

As the NTS-KE server embeds time duration information in the respective messages, its local time should be accurate to within a few seconds compared to the controlled PTP network(s). To avoid any dependencies, it should synchronize to a secure external time source, for example an NTS-secured NTP server. The time information is also necessary to check the lifetime of certificates used.

### 2.5.4. Certificates

The authentication of the TLS communication parties is based on certificates issued by a trusted Certificate Authority (CA) that are utilized during the TLS handshake. In classical TLS applications only servers are required to have them. For the key management system described here, the PTP nodes also need certificates to allow only authorized and trusted devices to get the group key and join a secure PTP network. (As TLS only authenticates the communication partners, authorization has to be managed by external means, see the topic Authorization in Section 2.5.5.4.) The verification of a certificate always requires a loose time synchronicity, because they have a validity period. This, however, reveals the well-known start-up problem, since secure time transfer itself requires valid certificates. (See the discussion and proposals on this topic in IETF RFC 8915 [RFC8915], Section 8.5 Initial Verification of Server certificates which applies to client and server certificates in the PTP key management system, too.)

Furthermore, some kind of Public Key Infrastructure (PKI) is necessary, which may be conceivable via the Online Certificate Status Protocol (OCSP, IETF RFC 6960) [RFC6960] as well as offline via root CA certificates.

The TLS communication parties must be equipped with a private key and a certificate in advance. The certificate contains a digital signature of the CA as well as the public key of the sender. The key pair is required to establish an authenticated and encrypted channel for the initial TLS phase. Distribution and update of the certificates can be done manually or automatically. However, it is important that they are issued by a trusted CA instance, which can be either local (private CA) or external (public CA).

For the certificates the standard for X.509 [ITU-T\_X.509] certificates MUST be used. Additional data in the certificates like domain, sdoId and/or subgroup attributes may help in authorizing. In that case it should be noted that using the PTP device in another network then implies to have a new certificate, too. Working with certificates without authorization information would not have that disadvantage, but more configuring at the NTS-KE server would be necessary: which domain, sdoId and/or subgroup attributes belong to which certificate.

As TLS is used to secure both sub-protocols, the NTS-KE and the NTS-TSR protocol, a comment on the security of TLS seems reasonable. A TLS 1.3 connection is considered secure today. However, note that a DoS (Denial of Service) attack on the key server can prevent new connections or parameter updates for secure PTP communication. A hijacked key management system is also critical, because it can completely disable the protection mechanism. A redundant implementation of the key server is therefore essential for a robust system. A further mitigation can be the limitation of the number of TLS requests of single PTP nodes to prevent flooding. But such measures are out of the scope of this document.

#### 2.5.5. Upfront Configuration

All PTP instances as well as the NTS-KE server need to be configured by the network administrator. This applies to several fields of parameters.

##### 2.5.5.1. Security Parameters

The cryptographic algorithm and associated parameters (the so-called security association(s) SA) used for PTP keys are configured by network operators at the NTS-KE server. PTP instances that do not support the configured algorithms cannot operate with the security. Since most PTP networks are managed by a single organization, configuring the cryptographic algorithm (MAC) for ICV calculation is practical. This prevents the need for the NTS-KE server and PTP instances to implement an NTS algorithm negotiation protocol.

For the ticket-based mode the AEAD algorithms need to be specified which the PTP grantors and the NTS-KE server support and negotiate during the registration process. Optionally, the MAC algorithm may be negotiated during a unicast PTP Key Request to allow faster or stronger algorithms, but a standard algorithm supported by every instance should be defined. Eventually, suitable algorithms may be defined in a respective PTP profile.

#### 2.5.5.2. Key Lifetimes

Supplementary to the above mentioned SAs the desired key rotation periods, i.e., the lifetimes of keys respectively all security parameters need to be configured at the NTS-KE server. This applies to the lifetime of a group key in the group-based mode as well as the lifetime of ticket key and unicast key in the ticket-based unicast mode (typically for every unicast pair in general). In addition, the corresponding update periods and grace periods need to be defined. Any particular lifetime, update period and grace period is configured as time spans specified in seconds.

#### 2.5.5.3. Certificates

The network administrator has to supply each PTP instance and the NTS-KE server with their X.509 certificates. The TLS communication parties must be pre-equipped with a private key and a certificate containing the public key (see Section 2.5.4).

#### 2.5.5.4. Authorization

The certificates provide authentication of the communication partners. Normally, they do not contain authorization information. Authorization decides, which PTP instances are allowed to join a group (in any of the group-based modes) or may enter a unicast communication in the ticket-based mode and request the respective SA(s) and key.

As mentioned, members of a group (multicast mode, mixed multicast/unicast mode) are identified by their domain and their sdoId. PTP domain and sdoId may be attributes in the certificates of the potential group members supplying additional authorization. If not contained in the certificates extra authorization means are necessary. (See also the discussion on advantages and disadvantages on certificates containing additional authorization data in Section 2.5.4.)

If the special Group-of-2 mode is used, the optional subGroup parameter (i.e., the subgroup number) needs to be specified at all members of respective Go2s, upfront. To enable the NTS-KE server to supply the subgroup members with the particular security data their respective certificates may reflect permission to take part in the subgroup. Else another authorization method is to be used.



In native unicast mode, any authenticated grantor that is (authorized) member of the group used for multicast may request a registration for unicast communication at the NTS-KE server (implicit authorization). If no group authorization is available (e.g., unicast only operation) another authentication scheme is necessary.

In the same way, any requester may request security data for a unicast connection with a specific grantor. Only authentication at the NTS-KE server using its certificate and membership in the group used for multicast is needed (implicit authorization). If a unicast communication is not desired by the grantor, it should not grant a specific unicast request. Again, if no group authorization is available (e.g., unicast only operation) another authentication scheme is necessary.

Authorization can be executed at least in some manual configuration. Probably the application of a standard access control system like Diameter, RADIUS or similar would be more appropriate. Also role-based access control (RBAC), attribute-based access control (ABAC) or more flexible tools like Open Policy Agent (OPA) could help administering larger systems. But details of the authorization of PTP instances lie out of scope of this document.

#### 2.5.5.5. Transparent Clocks

Transparent Clocks (TC) need to be supplied with respective certificates, too. For group-based modes they must be configured for the particular PTP domain and sdoId and eventually for the specific subgroup(s) when using Group-of-2. They need to request for the relevant group key(s) at the NTS-KE server to allow secure use of the correction field in a PTP message and generation of a corrected ICV. If TCs are used in ticket-based unicast mode, they need to be authorized for the particular unicast path.

Authorization of TCs for the respective groups, subgroups and unicast connections is paramount. Otherwise the security can easily be broken with attackers pretending to be TCs in the path. Authorization of TCs is necessary too in unicast communication, even if the normal unicast partners need not be especially authorized.

Transparent clocks may notice that the communication runs secured. In the group-based modes multicast mode and mixed multicast/unicast mode they construct the GroupID from domain and sdoId and request a group key from the NTS-KE server. Similarly, they can use the additional subgroup attribute in Go2 mode for a (group) key request. Afterwards they can check the ICV of incoming messages, fill in the correction field and generate a new ICV for outgoing messages. In ticket-based unicast mode a TC may notice a secured unicast request

from a requester to the grantor and can request the unicast key from the NTS-KE server to make use of the correction field afterwards. As mentioned above upfront authentication and authorization of the particular TCs is paramount not to open the secured communication to attackers.

#### 2.5.5.6. Start-up considerations

At start-up of a single PTP instance or the complete PTP network some issues have to be considered.

At least loose time synchronization is necessary to allow for authentication using the certificates. See the discussion and proposals on this topic in IETF RFC 8915 [RFC8915], Section 8.5 Initial Verification of Server certificates which applies to client and server certificates in the PTP key management system, too.

Similarly, to a key re-request during an update period, key requests SHOULD be started at a random point in time after start-up to avoid peak load on the NTS-KE server. Every grantor must register with the NTS-KE server before requesters can request a unicast key (and ticket).

### 3. NTS Messages for PTP

This section describes the structure of the specific NTS messages for the PTP key management. Table 2 to Table 11 specify which records the messages are composed of. The Mode column indicates the intended use of the particular record for the respective PTP communication mode. The next column informs whether the respective record is mandatory or optional. The reference column in the tables refer to the specific subsections of the record specification. The right column shows typical values as an example.

More details especially on the records the messages are built of and their types, sizes, requirements and restrictions are given in Section 4.

The NTS messages MUST contain the records given for the particular message though not necessarily in the same sequence indicated. Only the End of Message record MUST be the final record.

## 3.1. PTP Key Request Message

Table 2 shows the record structure of a PTP Key Request message. The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. The following Association Mode record describes the mode how the PTP instance wants to communicate: In the group-based mode the desired group number (plus eventually the subgroup attribute) is given. For ticket-based unicast communication the Association Mode contains the identification of the desired grantor, for example IPv4 and its IP address.

\*PTP Key Request (NTS-KE protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Next Protocol Negotiation	GrM / TiM	mandatory	Section 4.2.8	PTPv2.1
Association Mode	GrM / TiM	mandatory	Section 4.2.2	(Association Type    Association Value)
Supported MAC Algorithms	TiM	optional	Section 4.2.14	CMAC
Source PortIdentity	TiM	mandatory	Section 4.2.12	{binary data}
End of Message	GrM / TiM	mandatory	Section 4.2.4	(no record body)

Table 2: Record structure of the PTP Key Request message

Only in TiM, an optional record may follow. It offers the possibility to choose from additional MAC algorithms and presents the supported algorithms from which the NTS-KE server may choose. Again, only in ticket-based unicast mode, the Source PortIdentity record gives the data of the identification of the applying requester, for example IPv4 and its IP address. The messages always end with an End of Message record.

3.2. PTP Key Response Message

Table 3 shows the record structure of a PTP Key Response message from the NTS-KE server (NTS-KE protocol). The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1.

\*PTP Key Response (NTS-KE protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Next Protocol Negotiation	GrM / TiM	mandatory	Section 4.2.8	PTPv2.1
Current Parameters	GrM / TiM	mandatory	Section 4.2.3	set of records {...}
Next Parameters	GrM / TiM	mandatory (only during update period)	Section 4.2.7	set of records {...}
End of Message	GrM / TiM	mandatory	Section 4.2.4	(no record body)

Table 3: Record structure of the PTP Key Response message.

The following Current Parameters record is a container record containing in separate records all the security data needed to join and communicate in the secured PTP communication during the current validity period. Table 5 shows the records contained in that container record, again with exemplary contents in the right column. For more details on the records contained in the Current Parameters container record see Section 4.2.3.

If the request lies inside the update period, a Next Parameters container record is additionally appended in the PTP Key Response message giving all the security data needed in the upcoming validity period. Its structure follows the same composition as the Current Parameters container record. (If that specific client is to be excluded from the group in the upcoming SA period no Next Parameters container SHALL be sent.) In case of an error, both parameters container records are removed and a single error record is inserted (see Table 4). The messages always end with an End of Message record.

\*PTP Key Response with Error (NTS-KE protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Next Protocol Negotiation	GrM / TiM	mandatory	Section 4.2.8	PTPv2.1
Error	GrM / TiM	mandatory	Section 4.2.5	Not authorized
End of Message	GrM / TiM	mandatory	Section 4.2.4	(no record body)

Table 4: Record structure of the PTP Key Response message in case of an error.

The structure of the respective container records (Current Parameters and Next Parameters) used in the PTP Key Response message is given below:

\*Current/Next Parameters container - PTP Key Response (NTS-KE protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
Security Association	GrM / TiM	mandatory	Section 4.2.11	data set {...}
Validity	GrM /	mandatory	Section	{1560s  300s  3s}

Period	TiM		4.2.18	
PTP Time Server	TiM	mandatory	Section 4.2.10	data set {...}
Ticket	TiM	mandatory	Section 4.2.15	data set {...}

Table 5: Record structure of the container records

### 3.3. PTP Registration Request Message

The PTP Registration Request message (NTS-TSR protocol) starts with the NTS Message Type record containing the message type as well as the message version number, here always 1.0, see Table 6. (As the message belongs to the NTS-TSR protocol, no NTS Next Protocol Negotiation record is necessary.)

\*PTP Registration Request (NTS-TSR protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Message Type	TiM	mandatory	Section 4.2.9	PTP Registration Request    v1.0
PTP Time Server	TiM	mandatory	Section 4.2.10	data set {...}
AEAD Algorithm Negotiation	TiM	mandatory	Section 4.2.1	{AEAD_512    AEAD_256}
Supported MAC Algorithms	TiM	mandatory	Section 4.2.14	{CMAC    HMAC}
End of Message	TiM	mandatory	Section 4.2.4	(no record body)

Table 6: Record structure of the PTP Registration Request message

The PTP Time Server record presents all known network addresses of this grantor that are supported for a unicast connection. The following AEAD Algorithm Negotiation record indicates which algorithms for encryption of the ticket the grantor supports.

Then the next record (not optional as in PTP Key Request) follows, presenting all the grantor's supported MAC algorithms. The Supported MAC Algorithms record contains a list and comprises the MAC algorithms supported by the grantor that are feasible for calculating the ICV when securing the PTP messages in TiM. The message always ends with an End of Message record.

### 3.4. PTP Registration Response Message

The PTP Registration Response message (NTS-TSR protocol) from the NTS-KE server starts with the NTS Message Type record containing the message type as well as the message version number, here always 1.0, see Table 7. (As the message belongs to the NTS-TSR protocol, no NTS Next Protocol Negotiation record is necessary.)

\*PTP Registration Response (NTS-TSR protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Message Type	TiM	mandatory	Section 4.2.9	PTP Registration Response    v1.0
Current Parameters	TiM	mandatory	Section 4.2.3	set of records {...}
Next Parameters	TiM	mandatory (only during update period)	Section 4.2.7	set of records {...}
Heartbeat Timeout	TiM	optional	Section 4.2.6	900s
End of Message	TiM	mandatory	Section 4.2.4	(no record body)

Table 7: Record structure of the PTP Registration Response message.

As in the NTS-KE protocol, the following Current Parameters record is a container record containing in separate records all the necessary parameters for the current validity period. Table 9 shows the records contained in that container record, again with exemplary contents in the right column. For more details on the records contained in the Current Parameters container record see Section 4.2.3.

\*PTP Registration Response with error (NTS-TSR protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Message Type	TiM	mandatory	Section 4.2.9	PTP Registration Response    v1.0
Error	TiM	mandatory	Section 4.2.5	Not authorized
End of Message	TiM	mandatory	Section 4.2.4	(no record body)

Table 8: Record structure of the PTP Registration Response message in case of an error.

The structure of the respective container records (Current Parameters and Next Parameters) used in the PTP Registration Response message is given below:

\*Current/Next Parameters container - PTP Registration Response (NTS-TSR protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
AEAD Algorithm Negotiation	TiM	mandatory	Section 4.2.1	AEAD_AES_SIV_CMAC_512
Validity Period	TiM	mandatory	Section 4.2.18	{2460s    400s    3s}
Ticket Key ID	TiM	mandatory	Section 4.2.17	278



Ticket Key	TiM	mandatory	Section 4.2.16	{binary data}
------------	-----	-----------	-------------------	---------------

Table 9: Record structure of the container records in the PTP Registration Response message

If the registration request lies inside the update period a Next Parameters container record is additionally appended giving all the security data needed in the upcoming validity period. Its structure follows the same composition as the Current Parameters container record. (If the respective grantor has not registered yet for the upcoming SA period or has revoked its service no Next Parameters container will be sent.) In case of an error, both parameters container records are removed and a single error record is inserted (see Table 8). The messages always end with an End of Message record.

### 3.5. PTP Registration Revoke Message

The PTP Registration Revoke message (NTS-TSR protocol) from the grantor starts with the NTS Message Type record containing the message type as well as the message version number, here always 1.0, see Table 10. (As the message belongs to the NTS-TSR protocol, no NTS Next Protocol Negotiation record is necessary.)

\*PTP Registration Revoke (NTS-TSR protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Message Type	TiM	mandatory	Section 4.2.9	PTP Registration Revoke    v1.0
Source PortIdentity	TiM	mandatory	Section 4.2.12	{binary data}
End of Message	TiM	mandatory	Section 4.2.4	(no record body)

Table 10: Record structure of the PTP Registration Revoke message

The second record contains the Source PortIdentity which identifies the grantor wanting to stop its unicast support. This allows the NTS-KE server to uniquely identify the grantor if the PTP device communicates with the NTS-KE server via a management port running multiple grantors. The message always ends with an End of Message record.

### 3.6. PTP Heartbeat Message

The PTP Heartbeat message (NTS-TSR protocol) from the grantor to the NTS-KE server starts with the NTS Message Type record containing the message type as well as the message version number, here always 1.0, see Table 11. (As the message belongs to the NTS-TSR protocol, no NTS Next Protocol Negotiation record is necessary.)

\*PTP Heartbeat Message (NTS-TSR protocol)\*

NTS Record Name	Mode	Use	Reference	Exemplary body contents
NTS Message Type	TiM	mandatory	Section 4.2.9	PTP Heartbeat    v1.0
Status	TiM	optional	Section 4.2.13	{grantor load    15}
End of Message	TiM	mandatory	Section 4.2.4	(no record body)

Table 11: Record structure of the PTP Heartbeat message in the NTS-TSR protocol

The second record contains the optional Status record which allows the grantor to present various status updates to the NTS-KE server. The message always ends with an End of Message record.

PTP Heartbeat messages provide grantors with the ability to send messages to the NTS-KE server at regular intervals to signal their own functionality. These messages can optionally also contain one or multiple status records (see Table 11), for example to improve load balancing between the registered time servers or to provide additional monitoring. The NTS-KE server MUST accept PTP Heartbeat messages from a grantor if they have been previously requested by the NTS-KE server in the Registration Response message. However, the NTS-KE server MAY discard heartbeat messages if they arrive more frequently than specified by the heartbeat timeout (see

Section 4.2.6). If the NTS-KE server receives heartbeat messages from a grantor even though this is not requested, the NTS-KE server SHOULD discard these messages and not process them further. Processing of the status information is optional and the status records MAY be ignored by the NTS-KE server. If the grantor sends heartbeat messages to the NTS-KE server, the frames SHOULD NOT exceed the maximum transmission unit (MTU, 1500 octets for Ethernet).

#### 4. NTS Records for PTP

The sections above described the principle communication sequences and structure for the new NTS messages. All messages follow the NTS Key Establishment Process stated in the first part (until the description of Figure 3 starts) of Section 4 of IETF RFC 8915 [RFC8915] with the exception that registration requests use the ALPN "ntstsr/1" and do not contain a Next Protocol Negotiation record:

\_The NTS key establishment protocol is conducted via TCP port 4460. The two endpoints carry out a TLS handshake in conformance with Section 3, with the client offering (via an ALPN extension\_[RFC7301]), and the server accepting, an application-layer protocol of "ntske/1". Immediately following a successful handshake, the client SHALL send a single request as Application Data encapsulated in the TLS-protected channel. Then, the server SHALL send a single response. After sending their respective request and response, the client and server SHALL send TLS "close\_notify" alerts in accordance with Section 6.1 of RFC 8446\_[RFC8446].

\_The client's request and the server's response each SHALL consist of a sequence of records formatted according to\_ Figure 7\_. The request and a non-error response each SHALL include exactly one NTS Next Protocol Negotiation record. The sequence SHALL be terminated by a "End of Message" record. The requirement that all NTS-KE messages be terminated by an End of Message record makes them self-delimiting.\_

\_Clients and servers MAY enforce length limits on requests and responses, however, servers MUST accept requests of at least 1024 octets and clients SHOULD accept responses of at least 65536 octets.\_

\_The fields of an NTS-KE record are defined as follows:\_"

- \_C (Critical Bit): Determines the disposition of unrecognized Record Types. Implementations which receive a record with an unrecognized Record Type MUST ignore the record if the Critical Bit is 0 and MUST treat it as an error if the Critical Bit is 1 (see Section 4.1.3).\_"

- Record Type Number: A 15-bit integer in network byte order. The semantics of record types 07 are specified in this memo. Additional type numbers SHALL be tracked through the IANA Network Time Security Key Establishment Record Types registry.
  - Body Length: The length of the Record Body field, in octets, as a 16-bit integer in network byte order. Record bodies MAY have any representable length and need not be aligned to a word boundary.
  - Record Body: The syntax and semantics of this field SHALL be determined by the Record Type.
- For clarity regarding bit-endianness: the Critical Bit is the most-significant bit of the first octet. In the C programming language, given a network buffer 'unsigned char b[]' containing an NTS-KE record, the critical bit is 'b[0] >> 7' while the record type is '((b[0] & 0x7f) << 8) + b[1]'.

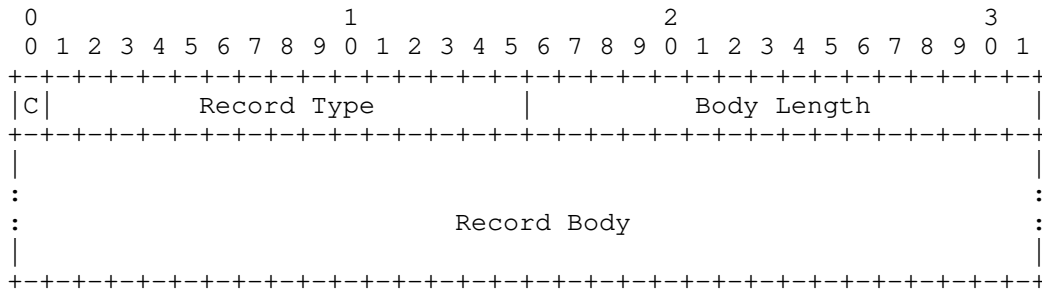


Figure 7: NTS-KE record format

Thus, all NTS messages consist of a sequence of records, each containing a Critical Bit C, the Record Type, the Body Length and the Record Body, see Figure 7. More details on record structure as well as the specific records used here are given in this section and respective subsections. So-called container records (short: container) themselves comprise a set of records in the record body that serve a specific purpose, e.g., the Current Parameters container record.

The records contained in a message may follow in arbitrary sequence (though nothing speaks against using the sequence given in the record descriptions), only the End of Message record MUST be the last one in the sequence indicating the end of the current message. Container records do not include an End of Message record.

## 4.1. Overview of the NTS Records

In Table 12 below, this section lists all NTS records the messages are built from. In addition to the NTS records already defined for NTP in IETF RFC 8915 (compare [RFC8915], Section 7.6. Network Time Security Key Establishment Record Types Registry), further records are required and begin with Record Type 1024 (needs to be defined by the IANA). The detailed structure and respective content of the records is given in Section 4.2. Besides the record number the sub-protocol it is used with, Table 12 indicates where to find it in [RFC8915] respectively in this document.

NTS Record Types	Description	Record Used in Protocol	Reference
0	End of Message	NTS-KE / NTS-TSR	[RFC8915] in Section 4.1.1. This document in Section 4.2.4
1	NTS Next Protocol Negotiation	NTS-KE	[RFC8915] in Section 4.1.2. This document in Section 4.2.8
2	Error	NTS-KE / NTS-TSR	[RFC8915] in Section 4.1.3. This document in Section 4.2.5
3	Warning	NTS-KE	[RFC8915] in Section 4.1.4. Not used for PTP
4	AEAD Algorithm Negotiation	NTS-TSR	[RFC8915] in Section 4.1.5. This document in Section 4.2.1
5	New Cookie for NTPv4	NTS-KE	[RFC8915] in Section 4.1.6. Not used for PTP
6	NTPv4 Server Negotiation	NTS-KE	[RFC8915] in Section 4.1.7. Not used for PTP
7	NTPv4 Port Negotiation	NTS-KE	[RFC8915] in Section 4.1.8. Not used for PTP

8 - 1023	Reserved for NTP		
1024	Association Mode	NTS-KE	Section 4.2.2
1025	Current Parameters	NTS-KE / NTS-TSR	Section 4.2.3
1026	Heartbeat Timeout	NTS-TSR	Section 4.2.6
1027	Next Parameters	NTS-KE / NTS-TSR	Section 4.2.7
1028	NTS Message Type	NTS-TSR	Section 4.2.9
1029	PTP Time Server	NTS-KE / NTS-TSR	Section 4.2.10
1030	Security Association	NTS-KE	Section 4.2.11
1031	Source PortIdentity	NTS-KE / NTS-TSR	Section 4.2.12
1032	Status	NTS-TSR	Section 4.2.13
1033	Supported MAC Algorithms	NTS-KE / NTS-TSR	Section 4.2.14
1034	Ticket	NTS-TSR	Section 4.2.15
1035	Ticket Key	NTS-TSR	Section 4.2.16
1036	Ticket Key ID	NTS-TSR	Section 4.2.17
1037	Validity Period	NTS-KE / NTS-TSR	Section 4.2.18
1038 - 16383	Unassigned		
16384 - 32767	Reserved for Private or Experimental Use		[RFC8915]



Numeric ID	AEAD Algorithm	Use	Key Length (Octets)	Reference
15	AEAD_AES_SIV_CMAC_256	mand.	32	[RFC5297]
16	AEAD_AES_SIV_CMAC_384	opt.	48	[RFC5297]
17	AEAD_AES_SIV_CMAC_512	opt.	64	[RFC5297]

Table 13: Recommended AEAD algorithms

- \* In a PTP Registration Request message, this record MUST be contained exactly once.
- \* In that message at least the AEAD\_AES\_SIV\_CMAC\_256 algorithm MUST be included.
- \* If multiple AEAD algorithms are supported, the grantor SHOULD put the algorithm identifiers in descending priority in the Record Body.
- \* Strong algorithms with higher bit lengths SHOULD have higher priority.
- \* In a PTP Registration Response message, this record MUST be contained exactly once in the Current Parameters container record and exactly once in the Next Parameters container record.
- \* The Next Parameters container record MUST be present only during the update period.
- \* The NTS-KE server SHOULD choose the highest priority AEAD algorithm from the request message that grantor and NTS-KE server support.
- \* The NTS-KE server MAY ignore the priority and choose a different algorithm that grantor and NTS-KE server support.
- \* In a PTP Registration Response message, this record MUST contain exactly one AEAD algorithm.
- \* The selected algorithm MAY differ in the corresponding Current Parameters container record and Next Parameters container record.

#### 4.2.2. Association Mode

Used in NTS-KE protocol

This record enables the NTS-KE server to distinguish between a group based request (multicast, mixed multicast/unicast, Group-of-2) or a unicast request. A multicast request carries a group number, while a unicast request contains an identification attribute of the grantor (e.g., IP address or PortIdentity).

Content and conditions:



- \* In a PTP Key Request message, this record MUST be contained exactly once.
- \* The record has a Record Type number of 1024 and the Critical Bit MAY be set.
- \* The Record Body SHALL consist of one association tuple:

Field	Octets	Offset
Association Type	2	0
Association Value	A	2

Table 14: Association Mode record

- \* The Association Type is a 16-bit unsigned integer.
- \* The length of Association Value depends on the value of Association Type.
- \* All data in the fields are stored in network byte order.
- \* The type numbers of Association Type (tbd. by IANA) as well as the length and content of Association Value are shown in the following table and more details are given below.

Description	Assoc. Type Number	Association Mode	Association Value Content	Assoc. Value Octets
Group	0	Multicast / Unicast*	Group Number	5
IPv4	1	Unicast	IPv4 address of the target port	4
IPv6	2	Unicast	IPv6 address of the target port	16
802.3	3	Unicast	MAC address of the target port	6
PortIdentity	4	Unicast	PortIdentity of the target PTP entity	10
	5 – 32767	Unassigned		
	32768 – 65565	Reserved for Private or Experimental Use		

Table 15: Association Types

Unicast\*: predefined groups of two (Group-of-2, Go2, see Group entry below)

Group:

- \* This Association Type allows a PTP instance to join a PTP multicast group.
- \* A group is identified by the PTP domain, the PTP profile (sdoId) and a sub-group attribute (see table below).
- \* The PTP domainNumber is an 8-bit unsigned integer in the closed range 0 to 255.

- \* The sdoId of a PTP domain is a 12-bit unsigned integer in the closed range 0 to 4095:
  - The most significant 4 bits are named the majorSdoId.
  - The least significant 8 bits are named the minorSdoId.
  - Reference: IEEE Std 1588-2019, Section 7.1.1
 \*sdoId = {majorSdoId || minorSdoId}\*
  
- \* The subGroup is 16-bit unsigned integer, which allows the division of a PTP multicast network into separate groups, each with individual security parameters.
- \* This also allows manually configured unicast connections (Group-of-2), which can include transparent clocks as well.
- \* The subGroup number is defined manually by the administrator.
- \* Access to the groups is controlled by authorization procedures of the PTP devices (see Section 2.5.5.4).
- \* If no subgroups are needed, this attribute MUST contain the value zero.
- \* The group number is eventually formed by concatenation of the following values:
 \*group number = {domainNumber || 4 bit zero padding || sdoId || subGroup}\*

This is equivalent to:

Bits 7 - 4	Bits 3 - 0	Octets	Offset
domainNumber (high)	domainNumber (low)	1	0
zero padding	majorSdoId	1	1
minorSdoId (high)	minorSdoId (low)	1	2
subGroup high octet (high)	subGroup high octet (low)	1	4
subGroup low octet (high)	subGroup low octet (low)	1	5

Table 16: Group Association

IPv4:

- \* This Association Type allows a requester to establish a PTP unicast connection to the desired grantor.
- \* The Association Value contains the IPv4 address of the target PTP entity.

- \* The total length is 4 octets.

#### IPv6:

- \* This Association Type allows a requester to establish a PTP unicast connection to the desired grantor.
- \* The Association Value contains the IPv6 address of the target PTP entity.
- \* The total length is 16 octets.

#### 802.3:

- \* This Association Type allows a requester to establish a PTP unicast connection to the desired grantor.
- \* The Association Value contains the MAC address of the Ethernet port of the target PTP entity.
- \* The total length is 6 octets.
- \* This method supports the IEEE 802.3 mode in PTP, where no UDP/IP stack is used.

#### PortIdentity:

- \* This Association Type allows a requester to establish a PTP unicast connection to the desired grantor.
- \* The Association Value contains the PortIdentity of the target PTP entity.
- \* The total length is 10 octets.
- \* The PortIdentity consists of the attributes clockIdentity and portNumber:  
\*PortIdentity = {clockIdentity || portNumber}\*  
\* The clockIdentity is an 8 octet array and the portNumber is a 16-bit unsigned integer.
- \* Source: IEEE Std 1588-2019, Sections 5.3.5 and 7.5

#### 4.2.3. Current Parameters

Used in NTS-KE and NTS-TSR protocol

This record is a simple container that can carry an arbitrary number of NTS records. It holds all security parameters relevant for the current validity period. The content as well as further conditions are defined by the respective NTS messages. The order of the included records is arbitrary and the parsing rules are so far identical with the NTS message. One exception: An End of Message record SHOULD NOT be present and MUST be ignored. When the parser reaches the end of the Record Body quantified by the Body Length, all embedded records have been processed.

Content and conditions:

- \* The record has a Record Type number of 1025 and the Critical Bit MAY be set.
- \* In a PTP Key Response message, this record MUST be contained exactly once.
- \* The Record Body is defined as a set of records and MAY contain the following records:

NTS Record Name	Communication Type	Use	Reference
Security Associations	Multicast / Unicast	mandatory	Section 4.2.11
Validity Period	Multicast / Unicast	mandatory	Section 4.2.18
PTP Time Server	Unicast	mandatory	Section 4.2.10
Ticket	Unicast	mandatory	Section 4.2.15

Table 17: Current Parameters container for PTP Key Response message

- \* The records Security Association and Validity Period MUST be contained exactly once.
- \* Additionally, the records PTP Time Server and Ticket MUST be included exactly once if the client wants a unicast connection in TiM and MUST NOT be included if the client wants to join a multicast group in GrM.
- \* In a PTP Registration Response message, the Current Parameters container record MUST be contained exactly once.
- \* The Record Body MUST contain the following records exactly:

NTS Record Name	Use	Reference
AEAD Algorithm Negotiation	mandatory	Section 4.2.1
Validity Period	mandatory	Section 4.2.18
Ticket Key ID	mandatory	Section 4.2.17
Ticket Key	mandatory	Section 4.2.16

Table 18: Current Parameters container for PTP  
Registration Response Message

#### 4.2.4. End of Message

Used in NTS-KE and NTS-TSR protocol

The End of Message record is defined in IETF RFC 8915 [RFC8915], Section 4:

"The record sequence in an NTS message SHALL be terminated by an End of Message record. The requirement that all NTS-KE messages be terminated by an End of Message record makes them self-delimiting."

Content and conditions:

- \* The record has a Record Type number of 0 and a zero-length body.
- \* The Critical Bit MUST be set.
- \* This record MUST occur exactly once as the final record of every NTS message.
- \* This record SHOULD NOT be included in the container records and MUST be ignored if present.
- \* See also: IETF RFC 8915, Section 4.1.1

#### 4.2.5. Error

Used in NTS-KE and NTS-TSR protocol

The Error record is defined in IETF RFC 8915 [RFC8915], Section 4.1.3. In addition to the Error codes 0 to 2 specified there the following Error codes 3 to 4 (tbd. by IANA) are shown below:

Error Code	Description
0	Unrecognized Critical Record
1	Bad Request
2	Internal Server Error
3	Not Authorized
4	Grantor not Registered
5 - 32767	Unassigned
32768 - 65535	Reserved for Private or Experimental Use

Table 19: Error Codes

Content and conditions:

- \* The record has a Record Type number of 2 and body length of two octets consisting of an unsigned 16-bit integer in network byte order, denoting an error code.
- \* The Critical Bit MUST be set.
- \* The Error code 3 "Not Authorized" is sent by the NTS-KE server if the requester is not authorized to join the desired multicast group or if a grantor is prohibited to register with the NTS-KE server.
- \* The Error record MUST NOT be included in a PTP Registration Request message.
- \* The Error code 4 "Grantor not Registered" is sent by the NTS-KE server when the requester wants to establish a unicast connection to a grantor that is not registered with the NTS-KE server.
- \* The Error record MUST NOT be included in a PTP Key Request message.

#### 4.2.6. Heartbeat Timeout

Used in NTS-TSR protocol

This record provides the NTS-KE server the capability to monitor the availability of the registered grantors. If this optional record is used, the registered grantors SHOULD send a PTP Heartbeat message to the NTS-KE server before the timeout expires.

Content and conditions:

- \* The record has a Record Type number of 1026 and the Critical Bit SHOULD NOT be set.
- \* The Record Body consists of a 16-bit unsigned integer in network byte order and denotes the heartbeat timeout in seconds.
- \* The timeout set by the NTS-KE server MUST NOT be less than 1s.
- \* The timeout starts at the NTS-KE server with the generation of the Registration Response message.
- \* Grantors that receive an invalid value as a heartbeat timeout MUST ignore this record and MUST NOT send heartbeat messages.
- \* Grantors that receive a valid value SHOULD send a PTP Heartbeat message to the NTS-KE server before the timeout has elapsed.
- \* The grantors SHOULD keep the heartbeat intervals and MAY also send heartbeat messages more frequently.
- \* After transmitting a heartbeat from the grantor to the NTS-KE server, both sides reset the timeout to the start value and let the time count down again.
- \* If this timeout is exceeded without receiving a heartbeat message or several heartbeats are missing in a row, the NTS-KE server MAY delete the grantor from its registration list, so that a new registration of the grantor is necessary.
- \* Grantors that are not (or no longer) registered with a NTS-KE server MUST NOT send heartbeat messages and NTS-KE servers MUST discard heartbeat messages from non-registered grantors.
- \* NTS-KE servers MAY respond in such cases with a Registration Response message containing error code 4 "Grantor not Registered".

#### 4.2.7. Next Parameters

Used in NTS-KE and NTS-TSR protocol

This record is a simple container that can carry an arbitrary number of NTS records. It holds all security parameters relevant for the upcoming validity period. The content as well as further conditions are defined by the respective NTS messages. The order of the included records is arbitrary and the parsing rules are so far identical with the NTS message. One exception: An End of Message record SHOULD NOT be present and MUST be ignored. When the parser reaches the end of the Record Body quantified by the Body Length, all embedded records have been processed.

Content and conditions:

- \* The record has a Record Type number of 1027 and the Critical Bit MAY be set.
- \* The Record Body is defined as a set of records.
- \* The structure of the record body and all conditions MUST be identical to the rules described in Section 4.2.3 of this document.



- \* In both the PTP Key Response and PTP Registration Response message, this record MUST be contained exactly once during the update period.
- \* Outside the update period, this record MUST NOT be included.
- \* In GrM, this record MAY also be missing if the requesting client is to be explicitly excluded from a multicast group after the security parameter rotation process by the NTS-KE server.
- \* More details are described in Section 2.5.1.

#### 4.2.8. NTS Next Protocol Negotiation

Used in NTS-KE protocol

The Next Protocol Negotiation record is defined in IETF RFC 8915 [RFC8915], Section 4.1.2:

\_"The Protocol IDs listed in the client's NTS Next Protocol Negotiation record denote those protocols that the client wishes to speak using the key material established through this NTS-KE server session. Protocol IDs listed in the NTS-KE server's response MUST comprise a subset of those listed in the request and denote those protocols that the NTP server is willing and able to speak using the key material established through this NTS-KE server session. The client MAY proceed with one or more of them. The request MUST list at least one protocol, but the response MAY be empty."\_

Content and conditions:

- \* The record has a Record Type number of 1 and the Critical Bit MUST be set.
- \* The Record Body consists of a sequence of 16-bit unsigned integers in network byte order.  
\*Record body = {Protocol ID 1 || Protocol ID 2 || }\*
- \* Each integer represents a Protocol ID from the IANA "Network Time Security Next Protocols" registry (tbd.) as shown in the table below.
- \* For NTS request messages for PTPv2.1 (NTS-KE protocol merely), only the Protocol ID for PTPv2.1 SHOULD be included.
- \* This prevents the mixing of records for different time protocols.

Protocol ID	Protocol Name	Reference
0	Network Time Protocol version 4 (NTPv4)	[RFC8915], Section 7.7
1	Precision Time Protocol version 2.1 (PTPv2.1)	This document
2 - 32767	Unassigned	
32768 - 65535	Reserved for Private or Experimental Use	

Table 20: NTS next protocol IDs

Possible NTP/PTP conflict:

- \* The support of multiple protocols in this record may lead to the problem that records in NTS messages can no longer be assigned to a specific time protocol.
- \* For example, an NTS request could include records for both NTP and PTP.
- \* However, NTS for NTP does not use NTS message types and the End of Message record is also not defined for the case of multiple NTS requests in one TLS message.
- \* This leads to the mixing of the records in the NTS messages.
- \* A countermeasure is the use of only a single time protocol in the NTS Next Protocol Negotiation record that explicitly assigns the NTS message to a specific time protocol.
- \* When using NTS-secured NTP and NTS-secured PTP, two separate NTS requests i.e., two separate TLS sessions MUST be made.

#### 4.2.9. NTS Message Type

Used in NTS-TSR protocol

This record enables the distinction between different NTS message types and message versions for the NTS-TSR protocol. It MUST be included exactly once in each NTS message in the NTS-TSR protocol.

Content and conditions:

- \* The record has a Record Type number of 1028 and the Critical Bit MUST be set.
- \* The Record Body MUST consist of three data fields:

Field		Octets	Offset
Message Type		2	0
Message Version	Major version	1	2
Message Version (cont.)	Minor version	1	3

Table 21: Content of the NTS Message Type record

- \* The Message Type field is a 16-bit unsigned integer in network byte order, denoting the type of the current NTS message.
- \* The following values (tbd. by IANA) are defined for the Message Type:

Message Type (value)	NTS Message (NTS-TSR protocol)
0	PTP Registration Request
1	PTP Registration Response
2	PTP Registration Revoke
3	PTP Heartbeat
4 - 32767	Unassigned
32768 - 65535	Reserved for Private or Experimental Use

Table 22: NTS Message Types for the NTS-TSR protocol

- \* The Message Version consists of a tuple of two 8-bit unsigned integers in network byte order:  
\*NTS Message Version = {major version || minor version}\*
  - \* The representable version is therefore in the range 0.0 to 255.255 (e.g., v1.4 = 0104h).
  - \* All NTS messages for PTPv2.1 described in this document are in version number 1.0.
  - \* Thus the Message Version MUST match 0100h.

#### 4.2.10. PTP Time Server

Used in NTS-KE and NTS-TSR protocol

The PTP Time Server record is used exclusively in TiM (PTP unicast connection) and signals to the client (PTP requester) for which grantor the security parameters are valid. This record is used both, in the NTS-KE protocol in the PTP Key Response, and in NTS-TSR protocol in the PTP Registration Request message.

Content and conditions:

- \* The record has a Record Type number of 1029 and the Critical Bit MAY be set.
- \* The record body consists of a set of association values constructed of the data tuple which forms the record body of the Association Mode record described in Section 4.2.2 (Association Mode).
- \* The structure of the record body and all conditions MUST be identical to the rules described in Section 4.2.2 (Association Mode) of this document, with the following exceptions:
- \* In a PTP Key Response message, this record MUST be contained exactly once within a container record (e.g., Current Parameters container record).
- \* The PTP Time Server record contains a list of all available addresses of the grantor assigned by the NTS-KE server.
- \* This MUST be one of the following association types: IPv4, IPv6, MAC address, as well as the PortIdentity of the grantor.
- \* It MUST NOT be the association type Group.
- \* The list SHALL contain only one of each association type.
- \* This allows the client to change the PTP transport mode (e.g., from IPv4 to IEEE 802.3) without performing a new NTS request.
- \* The list in the PTP Time Server record MUST NOT contain the Association Type number 0 (multicast group) and MUST contain at least one entry.
- \* The NTS-KE server SHOULD provide the grantor addresses requested by the client in the PTP Key Request message, but MAY also assign a different grantor to the client.
  
- \* In a PTP Registration Request message, this record MUST be included exactly once.
- \* The grantor MUST enter all network addresses that are supported for a unicast connection.
- \* This can be an IPv4, IPv6, MAC address, as well as the PortIdentity.
- \* The list in the PTP Time Server record MUST NOT contain the Association Type number 0 (multicast group) and MUST contain at least the PortIdentity.
- \* The PortIdentity is especially needed by the NTS-KE server to identify the correct PTP instance (the grantor) in case of a PTP Registration Revoke message.

4.2.11. Security Association

Used in NTS-KE protocol

This record contains the information "how" specific PTP message types must be secured. It comprises all dynamic (negotiable) values necessary to construct the AUTHENTICATION TLV (IEEE Std 1588-2019, Section 16.14.3). Static values and flags, such as the secParamIndicator, are described in more detail in Section 7.

Content and conditions:

- \* The record has a Record Type number of 1030 and the Critical Bit MAY be set.
- \* The Record Body is a sequence of various parameters in network byte order and MUST be formatted according to the following table:

Field	Octets	Offset
Security Parameter Pointer	1	0
Integrity Algorithm Type	2	1
Key ID	4	3
Key Length	2	7
Key	K	9

Table 23: Security Association record

- \* In a PTP Key Response message, the Security Association record MUST be included exactly once in the Current Parameters container record and the Next Parameters container record.
- \* The Next Parameters container record MUST be present only during the update period.
- \* In TiM, the Security Association record MUST be included exactly once in the encrypted Ticket as well.

Security Parameter Pointer

- \* The Security Parameter Pointer (SPP) is an 8-bit unsigned integer in the closed range 0 to 255.
- \* This value enables the mutual assignment of SA, SP and AUTHENTICATION TLVs.

- \* The generation and management of the SPP is controlled by the NTS-KE server (see Section 5.3).

#### Integrity Algorithm Type

- \* This value is a 16-bit unsigned integer in network byte order.
- \* The possible values are equivalent to the MAC algorithm types from the table in Section 4.2.14.
- \* The value used depends on the negotiated or predefined MAC algorithm.

#### Key ID

- \* The key ID is a 32-bit unsigned integer in network byte order.
- \* The field length is oriented towards the structure of the AUTHENTICATION TLV.
- \* The generation and management of the key ID is controlled by the NTS-KE server.
- \* The NTS-KE server MUST ensure that every key ID is unique.
  - Previous key IDs SHOULD NOT be reused for a certain number of rotation periods or a defined period of time (see Section 5.3).

#### Key Length

- \* This value is a 16-bit unsigned integer in network byte order, denoting the length of the key in octets.

#### Key

- \* The value is a sequence of octets with a length of Key Length.
- \* This symmetric key is needed together with the MAC algorithm to calculate the ICV.
- \* It can be both a group key (GrM) or a unicast key (TiM).

#### 4.2.12. Source PortIdentity

Used in NTS-KE and NTS-TSR protocol

This record contains a PTP PortIdentity and serves as an identifier. In a PTP Key Request message, it enables the unique assignment of the NTS request to the PTP instance of the sender, since the request may have been sent to the NTS-KE server via a management port.

The PortIdentity is embedded in the PTP Key Response message within the ticket to bind it to the PTP requester. Grantors can verify that the ticket comes from the correct sender when it is received and before it is decrypted, to prevent possible crypto-performance attacks. In a PTP registration Revoke message this record enables

the assignment of the grantor at the NTS-KE server to revoke an existing registration. This is necessary because requesting PTP devices may have multiple independent PTP ports and possibly multiple registrations with the KE.

Content and conditions:

- \* The record has a Record Type number of 1031 and the Critical Bit MAY be set.
- \* The record contains the PTP PortIdentity of the sender in network byte order, with a total length of 10 octets.
- \* In a PTP Key Request message, this record MUST be included exactly once if the client intends a unicast request in TiM and MUST NOT be included if the client intends to join a multicast group/Go2 (= GrM).
- \* In a PTP Registration Revoke message, this record MUST be included exactly once.
- \* The PortIdentity consists of the attributes clockIdentity and portNumber:  
\*PortIdentity = {clockIdentity || portNumber}\*
- \* The clockIdentity is an 8-octet array and the portNumber is a 16-bit unsigned integer (source: [IEEE1588-2019], Sections 5.3.5 and 7.5)

#### 4.2.13. Status

Used in NTS-TSR protocol

The Status record is an optional record that represents the current load of the grantor if status type is 0. It allows the NTS-KE server to improve load balancing when assigning grantors to the requesting PTP clients in TiM. The content of the record is designed in such a way that it can also transmit other information (e.g., manufacturer-related information).

Content and conditions:

- \* The record has a Record Type number of 1032 and the Critical Bit SHOULD NOT be set.
- \* The Record Body MUST consist of two data fields:

Field	Octets	Offset
Status Type	2	0
Status Data	D	2

Table 24: Structure of the Status record

- \* The Status Type is a 16-bit unsigned integer in network byte order, denoting the content of the Status Data field.
- \* The Status Data field is a sequence of octets in network byte order whose length, content and structure is determined by the Status Type field.
- \* The following values are currently set:

Status Type	Status Data length	Description
0	1 octet (unsigned int)	grantor load
1 - 32767	Unassigned	
32768 - 65535	Reserved for Private or Experimental Use	

Table 25: Values for Status Data

- \* With Status Type 0 (tbd. by IANA) the value designates the grantor load percentage (0% - 100%).
- \* The NTS-KE server shall ignore values greater than 100.
- \* In a PTP Heartbeat message this record MAY be contained once or several times.
- \* If multiple status records are included, the status type MUST NOT occur twice.
- \* The NTS-KE server MAY use the status record for optimizations and MAY also ignore them.

#### 4.2.14. Supported MAC Algorithms

Used in NTS-KE and NTS-TSR protocol



This record allows free negotiation of the MAC algorithm needed to generate the ICV. Since multicast groups are restricted to a shared algorithm, this record is used mandatorily in a PTP Registration Request message and MAY be used (optionally) in a PTP Key Request message.

Content and conditions:

- \* The record has a Record Type number of 1033 and the Critical Bit MAY be set.
- \* The Record Body contains a sequence of 16-bit unsigned integers in network byte order.  
\*Supported MAC Algorithms = {MAC 1 || MAC 2 || }\*
- \* Each integer (tbd. by IANA) represents a MAC Algorithm Type defined in the table below.
- \* Duplicate identifiers SHOULD NOT be included.
- \* Each PTP node MUST support at least the HMAC-SHA256-128 algorithm.

MAC Algorithm Types	MAC Algorithm	ICV Length (octets)	Reference
0	HMAC-SHA256-128	16	[fiPS-PUB-198-1], [IEEE1588-2019]
1	HMAC-SHA256	32	[fiPS-PUB-198-1]
2	AES-CMAC	16	[RFC4493]
3	AES-GMAC-128	16	[RFC4543]
4	AES-GMAC-192	24	[RFC4543]
5	AES-GMAC-256	32	[RFC4543]
6 - 32767	Unassigned		
32768 - 65535	Reserved for Private or Experimental Use		

Table 26: MAC Algorithms

In Group-based mode (GrM):

- \* This record is not necessary, since all PTP nodes in a multicast group MUST support the same MAC algorithm.
- \* Therefore, this record SHOULD NOT be included in a PTP Key Request message and the NTS-KE server MUST ignore this record if the Association Type in the Association Mode record is 0 (= multicast group).
- \* Unless this is specified otherwise by a PTP profile, the HMAC-SHA256-128 algorithm SHALL be used by default.

In Ticket-based mode (TiM):

- \* In a PTP Key Request message, this record MAY be contained if the requester wants a unicast connection (TiM, not Go2) to a specific grantor.
- \* The requester MUST NOT send more than one record of this type.
- \* If this record is present, at least the HMAC-SHA256-128 MAC algorithm MUST be included.
- \* If multiple MAC algorithms are supported, the requester SHOULD put the desired algorithm identifiers in descending priority in the record body.
- \* Strong algorithms with higher bit lengths SHOULD have higher priority.
- \* In a PTP Registration Request message, this record MUST be present and the grantor MUST include all supported MAC algorithms in any order.
- \* The NTS-KE server selects the algorithm after receiving a PTP Key Request message in unicast mode.
- \* The NTS-KE server SHOULD choose the highest priority MAC algorithm from the request message that grantor and requester support.
- \* The NTS-KE server MAY ignore the priority and choose a different algorithm that grantor and requester support.
- \* If the MAC Algorithm Negotiation record is not within the PTP Key Request message, the NTS-KE server MUST choose the default algorithm HMAC-SHA256-128.

Initialization Vector (IV)

- \* If GMAC is to be supported as a MAC algorithm, then an Initialization Vector (IV) must be constructed according to IETF RFC 4543 [RFC4543], Section 3.1.
- \* Therefore, the IV MUST be eight octets long and MUST NOT be repeated for a specific key.
- \* This can be achieved, for example, by using a counter.

#### 4.2.15. Ticket

Used in NTS-KE protocol

This record contains the parameters of the negotiated AEAD algorithm chosen between the grantor and the NTS-KE server, as well as an encrypted security association. The record contains all the necessary security parameters that the grantor needs for a secured PTP unicast connection to the requester. The ticket is encrypted by the NTS-KE server with the symmetric ticket key which is also known to the grantor. The requester is not able to decrypt the encrypted security association within the ticket.

Content and conditions:

- \* The record has a Record Type number of 1034 and the Critical Bit MAY be set.
- \* The Record Body consists of several data fields and MUST be formatted as follows.

Field	Octets	Offset
Ticket Key ID	4	0
Source PortIdentity	10	4
Nonce Length	2	14
Nonce	N	16
Encrypted SA Length	2	N+16
Encrypted Security Association	E	N+18

Table 27: Structure of a Ticket record

- \* In a PTP Key Response message, this record MUST be included exactly once each in the Current Parameters container record and the Next Parameters container record if the requesting client wants a unicast communication to a specific grantor in TiM.
- \* The Next Parameters container record MUST be present only during the update period.

Ticket Key ID

- \* This is a 32-bit unsigned integer in network byte order, denoting the key ID of the ticket key.
- \* The value is set by the NTS-KE server and is valid for the respective validity period.
- \* See also Section 4.2.17 for more details.

## Source PortIdentity

- \* This 10-octet long field contains the identical Source PortIdentity of the PTP client from the PTP Key Request message.

## Nonce Length

- \* This is a 16-bit unsigned integer in network byte order, denoting the length of the Nonce field in octets.

## Nonce

- \* This field contains the Nonce needed for the AEAD operation.
- \* The length and conditions attached to the Nonce depend on the AEAD algorithm used.
- \* More details and conditions are described in Section 5.2.

## Encrypted SA Length

- \* This is a 16-bit unsigned integer in network byte order, denoting the length of the Encrypted Security Association field in octets.

## Encrypted Security Association

- \* This field contains the output of the AEAD operation (Ciphertext) after the encryption process of the respective Record Body of the respective Security Association record.
- \* The plaintext of this field is described in Section 4.2.11.
- \* More details about the AEAD process and the required input data are described in Section 5.2.

## 4.2.16. Ticket Key

Used in NTS-TSR protocol

This record contains the ticket key, which together with an AEAD algorithm is used to encrypt and decrypt the ticket payload (content of the Encrypted Security Association field in the Ticket record).

Content and conditions:

- \* The record has a Record Type number of 1035 and the Critical Bit MAY be set.
- \* The Record Body consists of a sequence of octets holding the symmetric key for the AEAD function.
- \* The generation and length of the key MUST meet the requirements of the associated AEAD algorithm.

- \* In a PTP Registration Response message, this record MUST be included exactly once each in the Current Parameters container record and the Next Parameters container record.
- \* The Next Parameters container record MUST be present only during the update period.

#### 4.2.17. Ticket Key ID

Used in NTS-TSR protocol

The Ticket Key ID record is a unique identifier that allows a grantor to identify the associated ticket key. The NTS-KE server is responsible for generating this key ID, which is also unique to the PTP network and incremented at each rotation period. The associated key is known only to the NTS-KE server and grantor, and is generated and exchanged during the registration phase of the grantor. All tickets generated by the NTS-KE server for the corresponding grantor in this validity period using the same ticket key ID.

Content and conditions:

- \* The record has a Record Type number of 1036 and the Critical Bit MAY be set.
- \* The Record Body consists of a 32-bit unsigned integer in network byte order.
- \* The generation and management of the ticket key ID is controlled by the NTS-KE server.
- \* The NTS-KE server must ensure that every ticket key has a unique number.
  - The value is implementation dependent and MAY be an enumeration.
  - Previous IDs SHOULD NOT be reused for a certain number of rotation periods or a defined period of time.
- \* In a PTP Registration Response message, this record MUST be included exactly once in the Current Parameters container record and once in the Next Parameters container record.
- \* The Next Parameters container record MUST be present only during the update period.
- \* The Ticket record MUST be present in TiM and MUST NOT be present in GrM.

#### 4.2.18. Validity Period

Used in NTS-KE and NTS-TSR protocol

This record contains the validity information of the respective security parameters (see also Section 2.5.1).

Content and conditions:

- \* In a PTP Key Response as well as in the PTP Registration Response message, this record MUST be included exactly once each in the Current Parameters container record and the Next Parameters container record.
- \* The record has a Record Type number of 1037 and the Critical Bit MAY be set.
- \* The Record Body MUST consist of three data fields:

Field	Octets	Offset
Lifetime	4	0
Update Period	4	4
Grace Period	4	8

Table 28: Structure of a Validity Period record

Lifetime

- \* The Lifetime is a 32-bit unsigned integer in network byte order.
- \* If this record is within a Current Parameters container record, it shows the remaining lifetime of the security parameters for the current validity period in seconds.
- \* If this record is within a Next Parameters container record, it shows the total lifetime of the security parameters for the next validity period in seconds.
- \* The counting down of the Next Parameters lifetime starts as soon as the remaining lifetime of the Current Parameters reaches 0s.
- \* The maximum value is set by the NTS-KE administrator or the PTP profile.
- \* In conjunction with a PTP unicast establishment in TiM, the lifetime of the unicast key (within the Security Association record), the ticket key and registration lifetime of a grantor with the NTS-KE server MUST be identical.

Update Period

- \* The Update Period is a 32-bit unsigned integer in network byte order.
- \* It specifies how many seconds before the lifetime expires the update period starts.

- \* Unlike the lifetime, this is a fixed value that is not counted down.
- \* The Update Period value MUST NOT be greater than the full Lifetime.
- \* Recommended is an Update Period of 120s-300s if the full Lifetime is 900s or longer.
- \* If the value of the Update Period in the Current Parameters container record is greater than the Lifetime, then the key update process has started.
- \* The presence or absence of the Next Parameters container record is specified in Section 4.2.7.

#### Grace Period

- \* The Grace Period is a 32-bit unsigned integer in network byte order.
- \* It defines how many seconds expired security parameters MUST still be accepted.
- \* This allows the verification of incoming PTP messages that were still on the network and secured with the old parameters.
- \* The Grace Period value MUST NOT be greater than the Update Period.
- \* Recommended is a Grace Period of 0 to 5 seconds.

#### Notes:

- \* Requests during the currently running lifetime will receive respectively adapted count values.
- \* The lifetime is a counter that is decremented and marks the expiration of defined parameters when the value reaches zero.
- \* The realization is implementation-dependent and can be done for example by a secondly decrementing.
- \* It MUST be ensured that jumps (e.g., by adjustment of the local clock) are avoided.
- \* The use of a monotonic clock is suitable for this.
- \* Furthermore, it is to be considered which consequences the drifting of the local clock can cause.
- \* With sufficiently small values of the lifetime (<12 hours), this factor should be negligible.

#### 5. Additional Mechanisms

This section provides information about the replay protection to be applied, the use of the negotiated AEAD algorithm as well as the generation of the security policy pointers.

## 5.1. Replay Protection

Protection against replay attacks is an important mechanism in NTS4PTP. Without it, attackers can distribute false timing information in the PTP network, especially in multicast communication mode, by replaying intercepted and authenticated PTP packets. For unicast connections, the lack of an anti-replay mechanism can lead to attackers breaking an existing contract of a requester or initiating a new one for it. Furthermore, attackers would be able to perform DoS amplification attacks using unicast contracts.

In principle, there are three ways to implement replay protection. The first option is to use a one-time-use random number that is embedded in a message as a unique identifier. In the case of bidirectional communication, such an identifier is included in a request, which the communication partner then copies into the response message. If the identifiers between request and response are identical, then a replay can be ruled out. The advantage is the statelessness of the responding communication partner, but this method can only be used with pure unicast connections (requestresponse principle).

The second option is to use absolute time information, such as that used by Kerberos [RFC4120] in its tickets. Messages have a defined lifetime and can no longer be misused. However, this requires time-synchronous devices and since PTP transmits timing information as payload data, the use of this variant is not possible.

The last one is the use of sequence numbers. Here, the sender increments a number contained in each message. The advantage is that both the packet sequence and possible packet loss can be identified. But this requires a stateful sender and possibly negotiation of the starting value. Furthermore, the range of values must not be too small, otherwise the protection may be broken if the overflow is not taken into account.

In NTS4PTP, the first variant is used for bidirectional communication (e.g., delay req/resp messages) whereas the third variant is used for unidirectional communication (e.g., sync messages). It does not matter whether the communication takes place in GrM or TiM. PTPv2.1 already uses a sequence ID in the PTP header to ensure the correct packet sequence, so that statelessness is not a goal. However, the sequence ID of two octets is too small to provide effective replay protection. With a maximum frame rate of 128 sync messages per second, an overflow already occurs after 8 min and 32 s. Even with unicast connections, this is not sufficient if the negotiated contract duration corresponds to the maximum duration of 1000s (16 m 40s) suggested by the PTP standard ([IEEE1588-2019], p. 374). For



this reason, NTS4PTP defines its own sequence numbers, which are handled individually for each PTP message type and for each PTP signaling message type. The sequence numbers have a size of four octets each, so that at the maximum frame rate an overflow only occurs after 388 days. A sequence number of a PTP message is stored in the corresponding SA as well as in the AuthTLV of a secured PTP message.

The mode of operation is based on the anti-replay algorithm defined for IPsec [RFC6479]. Replay protection therefore uses a sliding window whose size is configurable by the administrator. NTS4PTP initiates the sequence numbers with zero regardless of GrM or TiM and resets after each key rotation period to prevent the sequence numbers from overflowing. Since a new key is used to secure the PTP messages after resetting the sequence numbers, a replay attack is also prevented here. However, there is a special case with the random numbers (variant 1), which are used in bidirectional communication. These protect the PTP client from replay, but not the PTP server. This is problematic in unicast negotiation, because an attacker can negotiate new contracts at the grantor, which a PTP client did not want at all. NTS4PTP solves it with the grantor storing the random numbers from the unicast requests of a PTP client for the duration of the current validity period. Thus the grantor can check on incoming unicast requests whether the given number has already been used. Since the collision probability is very low when generating random numbers, the grantor could assume a replay attempt in such a case and discard the received message. The short-term buffering of numbers is not problematic from the current point of view, since only a few numbers need to be stored due to the short validity periods. Since this is only the case for authenticated messages, an attacker cannot fill the memory with bogus packets.

## 5.2. AEAD Operation

General information about AEAD:

- \* The AEAD operation enables the integrity protection and the optional encryption of the given data, depending on the input parameters.
- \* While the structure of the AEAD output after the securing operation is determined by the negotiated AEAD algorithm, it usually contains an authentication tag in addition to the actual ciphertext.
- \* The authentication tag provides the integrity protection, whereas the ciphertext represents the encrypted data.
- \* The AEAD algorithms supported in this document (see Section 4.2.1) always return an authentication tag with a fixed length of 16 octets.

- \* The size of the following ciphertext is equal to the length of the plaintext.
- \* The concatenation of authentication tag and ciphertext always form the unit Ciphertext:  
\*Ciphertext = {authentication tag || ciphertext}\*
- \* Hint: The term Ciphertext is distinguished between upper and lower case letters.
- \* The following text always describes "Ciphertext".
- \* Separation of the information concatenated in Ciphertext is not necessary at any time.
- \* Six parameters are relevant for the execution of an AEAD operation:
  - AEAD (...): is the AEAD algorithm itself
  - A: Associated Data
  - N: Nonce
  - K: Key
  - P: Plaintext
  - C: Ciphertext
- \* The protection and encryption of the data is done as follows: C = AEAD (A, N, K, P)
- \* Therefore, the output of the AEAD function is the Ciphertext.
- \* The verification and decryption of the data is done this way: P = AEAD (A, N, K, C)
- \* The output of the AEAD function is the Plaintext if the integrity verification is successful.

AEAD algorithm and input/output values for the Ticket record:

- \* AEAD ():
  - The AEAD algorithm that is negotiated between grantor and NTS-KE server during the registration phase.
  - A list of the AEAD algorithms considered in this document can be found in Section 4.2.1.
- \* Associated Data:
  - The Associated Data is an optional AEAD parameter and can be of any length and content, as long as the AEAD algorithm does not give any further restrictions.
  - In addition to the Plaintext, this associated data is also included in the integrity protection.
  - When encrypting or decrypting the Security Association record, this parameter MUST remain empty.
- \* Nonce:
  - Corresponds to the value from the Nonce field in the Ticket (Section 4.2.15).
  - The requirements and conditions depend on the selected AEAD algorithm.

- For the AEAD algorithms defined in Section 4.2.1 (with numeric identifiers 15, 16, 17), a cryptographically secure random number MUST be used.
- Due to the block length of the internal AES algorithm, the Nonce SHOULD have a length of 16 octets.
- \* Key:
  - This is the symmetric key required by the AEAD algorithm.
  - The key length depends on the selected algorithm.
  - When encrypting or decrypting the Security Association record, the ticket key MUST be used.
- \* Plaintext:
  - This parameter contains the data to be encrypted and secured.
  - For AEAD encryption, this corresponds to the Record Body of the Security Association record with all parameters inside.
  - This is also the output of the AEAD operation after the decryption process.
- \* Ciphertext:
  - Corresponds to the value from the Encrypted Security Association field in the Ticket (Section 4.2.15).
  - The Ciphertext is the output of the AEAD operation after the encryption process.
  - This is also the input parameter for the AEAD decryption operation.

### 5.3. SA/SPP Management

This section describes the requirements and recommendations attached to SA/SPP management, as well as details about the generation of identifiers.

Requirements for the Security Association Database management:

- \* The structure and management of the Security Association Database (SAD) are implementation-dependent both on the NTS-KE server and on the PTP devices.
- \* An example of this, as well as other recommendations, are described in Annex P of IEEE Std 1588-2019 ([IEEE1588-2019]).
- \* A PTP device MUST contain exactly one SAD and Security Policy Database (SPD).
- \* For multicast and Group-of-2 connections, SPPs MUST NOT occur more than once in the SAD of a PTP device.
- \* For unicast connections, SPPs MAY occur more than once in the SAD of a PTP device.
- \* The NTS-KE server MUST ensure that SPPs can be uniquely assigned to a multicast group or unicast connection.
- \* This concerns both the NTS-KE server and all PTP devices assigned to the NTS-KE server.

#### SPP generation:

The generation of the SPP always takes place on the NTS-KE server and enables the identification of a corresponding SA. The value of the SPP can be either a random number or an enumeration. An SPP used in any multicast group MUST NOT occur in any other multicast group or unicast connection. If a multicast group or unicast connection is removed by the NTS-KE server, the released SPPs MAY be reused for new groups or unicast connections. Before reusing an SPP, the NTS-KE server MUST ensure that the SPP is no longer in use in the PTP network (e.g., within Next Parameters). In different PTP devices, an SPP used in a unicast connection MAY also occur in another unicast connection, as long as they are not used in multicast groups.

#### Key/Key ID generation:

The generation of the keys MUST be performed by using a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) on the NTS-KE server (see also Section 2.5.2). The length of the keys depends on the MAC algorithm used. The generation and management of the key ID is also controlled by the NTS-KE server. The NTS-KE server MUST ensure that every key ID is unique at least within an SA with multiple parameter sets. The value of the key ID is implementation dependent and MAY be either a random number, a hash value or an enumeration. Key IDs of expired keys MAY be reused but SHOULD NOT be reused for a certain number of rotation periods or a defined period of time. Before reusing a key ID, the NTS-KE server MUST be ensured that the key ID is no longer in use in the PTP network (e.g., within Next Parameters).

#### 6. New TICKET TLV for PTP Messages

Once a PTP port is registered as a grantor for association in unicast mode another PTP port (requester) can associate with it by first requesting a key from the NTS-KE server with Association Type in the Association Mode record set to one of the values 1 to 4 (IPv4, IPv6, 802.3 or PortIdentity), and Association Values to the related address of the desired grantor. After the reception of a PTP Key Response message during the NTS-KE protocol the requester obtains the unicast key and the Ticket record containing the Record Body of the Security Association record (see Section 2.4 and Section 4.2.15). The ticket includes the identification of the requester, the Encrypted SA along with the unicast key as well as the lifetime in the Validity record.

To provide the grantor with the security data, the requester sends a secured unicast request to the grantor, e.g., an Announce request (= Signaling message with a REQUEST\_UNICAST\_TRANSMISSION TLV with Announce as messageType in the TLV), which is secured with the unicast key.

To accomplish that, the requester sends a newly defined TICKET TLV with the Ticket embedded and the AUTHENTICATION TLV with the PTP unicast negotiation message. The TICKET TLV must be positioned before the AUTHENTICATION TLV to include the TICKET TLV in the securing by the ICV. The receiving grantor decrypts the Ticket (actually the encrypted security association) from the TICKET TLV getting access to the information therein. With the contained unicast key, the grantor checks the requester identity and the authenticity of the request message.

Thereafter, all secured unicast messages between grantor and requester will use the unicast key for generating the ICV in the AUTHENTICATION TLV for authentication of the message until the unicast key expires.

If the requesters identity does not match with the Source PortIdentity field in the Ticket or the ICV in the AUTHENTICATION TLV is not identical to the generated ICV by the grantor, then the unicast request message MUST be denied.

The TICKET TLV structure is given in Table 29 below.

Field	Octets	Offset
tlvType	2	0
lengthfield	2	2
Ticket record	T	4

Table 29: Structure of the TICKET TLV

To comply with the TLV structure of IEEE Std 1588-2019 ([IEEE1588-2019], Section 14.1) the TICKET TLV is structured as presented in Table 29 with a newly defined tlvType, a respective length field and the Ticket record (see Section 4.2.15) containing the encrypted security association. Eventually the Ticket TLV may be defined externally to IEEE 1588 SA, e.g., by the IETF. Then the structure should follow IEEE Std 1588-2019 ([IEEE1588-2019], Section 14.3) to define a new standard organization extension TLV as presented in Table 30 below.

Field	Octets	Offset
tlvType	2	0
lengthfield	2	2
organizationId	3	4
organizationSubType	3	7
Ticket record	T	10

Table 30: Structure of an organization extension TLV form for the TICKET TLV

The TICKET TLV will be added to the PTP message preceding the AUTHENTICATION TLV as shown in figure 48 of IEEE Std 1588-2019 ([IEEE1588-2019], Section 16.14.1.1).

### 7. AUTHENTICATION TLV Parameters

The AUTHENTICATION TLV is the heart of the integrated security mechanism (prong A) for PTP. It provides all necessary data for the processing of the security means. The structure is shown in Table 31 below (compare to figure 49 of [IEEE1588-2019]).

TLV Field	Use	Description
tlvType	mandatory	TLV Type
lengthfield	mandatory	TLV Length Information
SPP	mandatory	Security Parameter Pointer
secParamIndicator	mandatory	Security Parameter Indicator
keyID	mandatory	Key Identifier or Current Key Disclosure Interval, depending on verification scheme
disclosedKey	optional	Disclosed key from previous interval
sequenceNo	optional	Sequence number
RES	optional	Reserved
ICV	mandatory	ICV based on algorithm OID

Table 31: Structure of the AUTHENTICATION TLV

The tlvType is AUTHENTICATION and lengthfield gives the length of the TLV. When using the AUTHENTICATION TLV with NTS key management, the SPP and keyID will be provided by the NTS-KE server in the PTP Key Response message

The optional disclosedKey, sequenceNo, and RES fields are omitted. So all of the flags in the SecParamIndicator MUST be FALSE.

ICV field contains the integrity check value of the particular PTP message calculated using the integrity algorithm defined by the key management.

(... Use of the field sequenceNo has to be discussed with the IEEE1588. If used it may enable replay protection for more than 388 days before an overflow occurs, see Section 5.1 ... )

## 8. IANA Considerations

Considerations should be made ...

...

## 9. Security Considerations

...

## 10. Acknowledgements

The authors would like to thank ...

## 11. References

### 11.1. Normative References

[fiPS-PUB-198-1]

National Institute of Standards and Technology (NIST),  
"The Keyed-Hash Message Authentication Code (HMAC)",  
NIST fiPS PUB 198-1, 2008.

[IANA\_AEAD]

Internet Assigned Numbers Authority (IANA, "Authenticated  
Encryption with Associated Data (AEAD) Parameters",  
IANA AEAD Parameters (2022), December 2022,  
<<https://www.iana.org/assignments/aead-parameters/aead-parameters.xhtml>>.

[IEEE1588-2019]

Institute of Electrical and Electronics Engineers - IEEE  
Standards Association, "IEEE Standard for a Precision  
Clock Synchronization Protocol for Networked Measurement  
and Control Systems", IEEE Standard 1588-2019, 2019.

[ITU-T\_X.509]

International Telecommunication Union (ITU), "Information  
technology Open systems interconnection The Directory:  
Public-key and attribute certificate frameworks", ITU-T  
Recommendation X.509 (2008), November 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The  
Kerberos Network Authentication Service (V5)", RFC 4120,  
DOI 10.17487/RFC4120, July 2005,  
<<https://www.rfc-editor.org/info/rfc4120>>.



- [RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <<https://www.rfc-editor.org/info/rfc4543>>.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <<https://www.rfc-editor.org/info/rfc5116>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<https://www.rfc-editor.org/info/rfc7301>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/info/rfc8915>>.

## 11.2. Informative References

[Langer\_et\_al.\_2020]

Langer, M., Heine, K., Sibold, D., and R. Bermbach, "A Network Time Security Based Automatic Key Management for PTPv2.1", 2020 IEEE 45th Conference on Local Computer Networks (LCN), Sydney, Australia, DOI 10.1109/LCN48667.2020.9314809, November 2020, <<https://ieeexplore.ieee.org/document/9314809>>.

[Langer\_et\_al.\_2022]

Langer, M. and R. Bermbach, "A comprehensive key management solution for PTP networks", Computer Networks, Volume 213 (2022), 109075, DOI 10.1016/j.comnet.2022.109075, June 2022, <<https://www.sciencedirect.com/science/article/pii/S1389128622002158>>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <<https://www.rfc-editor.org/info/rfc4493>>.

[RFC5297] Harkins, D., "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)", RFC 5297, DOI 10.17487/RFC5297, October 2008, <<https://www.rfc-editor.org/info/rfc5297>>.

[RFC6479] Zhang, X. and T. Tsou, "IPsec Anti-Replay Algorithm without Bit Shifting", RFC 6479, DOI 10.17487/RFC6479, January 2012, <<https://www.rfc-editor.org/info/rfc6479>>.

#### Authors' Addresses

Martin Langer  
Physikalisch-Technische Bundesanstalt (PTB)  
Bundesallee 100  
38116 Braunschweig  
Germany  
Email: martin.langer@ptb.de

Rainer Bermbach  
Ostfalia University of Applied Sciences  
Salzdahlumer Straße 46/48  
38302 Wolfenbüttel  
Germany  
Email: r.bermbach@ostfalia.de