Network Time Protocol
Internet-Draft
Intended status: Informational
Expires: July 2, 2021

J. Gruessing

December 29, 2020

NTPv5 use cases and requirements
draft-gruessing-ntp-ntpv5-requirements-01

Abstract

This document describes the use cases, requirements, and considerations that should be factored in the design of a successor protocol to supercede version 4 of the NTP protocol [RFC5905] presently referred to as NTP version 5 ("NTPv5"). This document is non-exhaustive and does not in its current version represent working group consensus.

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/I-D/tree/main/draft-gruessing-ntp-ntpv5-requirements [1].

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 July 2, 2021.

Copyright Notice

Table of Contents

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 also fallen victim to vulnerabilities that have made it used for distributed denial of service (DDoS) amplification attacks.

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.

2. Use cases and existing deployments of NTP

There are several common scenarios for exxisting NTPv4 deployments; publicly accessible NTP services such as the NTP Pool [ntppool] are used to offer clock synchronisation for end users and embedded devices, ISP provided servers to synchronise devices such as customer-premesis equipment where reduced accuracy may be tollerable. 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 also have deployed and offer NTP services both for internal use and for customers, particularly where the network is unable to offer or does not require PTP [IEEE-1588-2008]. 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.

3. Requirements

At a high level, NTPv5 should be a protocol that is capable of operating in both local networks and also over public internet connections where packet loss, delay, and even filtering may occur.

Timestamp resolution SHOULD either match or exceed NTPv4, and be extensible to represent any specified timescale.

The protocol SHOULD NOT transmit time zone information and should focus on providing clock synchronisation as TZDIST [RFC7808] already provides this ability.

3.1. IP affinity

Servers SHOULD have a new identifier that peers use as reference, this SHOULD NOT be a FQDN, an IP address, or identifier tied to a certificate. Servers SHOULD be able to migrate and change their identifiers as stratum topologies or network configuration changes occur.

Clients SHOULD re-establish connections with servers at an interval to prevent attempting to maintain connectivity to dead host and give network operators the ability to move traffic away from IP addresses in a timely manner. This functionality should also compliment having a "Kiss of Death" or similar message from servers.

3.2. Algorithms

Algorithms describing functions such as clock filtering, selection and clustering SHOULD be omitted from the specification; the specification should instead only provide only what is necessary to describe protocol semantics and normative behaviours.

The working group should consider creating a separate informational document to describe an algorithm to assist with implementation, and to consider adopting future documents which describe new algorithms as they are developed. Specifying client algorithms separately from the protocol allows will allow NTPv5 to meet the needs of applications with a variety of network properties and performance requirements. It also allows for innovation in implementations without sacrificing basic interoperability.

3.3. Timescales

Support SHOULD be available for other timescales in addition to UTC - this should include, but not limited to the use of TAI or Modified Julian Date as defined in [I-D.ietf-ntp-roughtime], however UTC SHALL be the default timescale and MUST be supported by all implementations. Consideration should be made to include listing the supported timescales either as part of specific IANA parameter registry, or as part of the extension registry.

3.4. Leap seconds

The specification or the protocol SHOULD be explicit about when a leap second is being applied, and the protocol should allow for transmiting an upcoming leap second ahead of the day it is to be applicable. Nevertheless, due to network delays and the polling interval, applications with NTP clients will need to manage the leap second event at their local clock.

3.4.1. Leap second smearing

Server responses SHOULD include not only an indicator as to wether the server supports smearing, but also if the current time being transmitted is smeared. The protocol may also transmit the start/end or duration of the smearing ahead of time. It MUST be possible for clients to determine the unsmeared time of the timescale.

3.5. Backwards compatibility to NTS and NTPv4

The support for compatibility with other protocols SHOULD NOT prevent addressing issues that have previous caused issues in deployments or cause ossification of the protocol. Protocol ossification MUST be

addressed to prevent existing NTPv4 deployments which incorrectly respond 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.

The model for backward compatibility is servers that support mutliple versions NTP and send a response in the same version as the request. This does not preclude high stratum servers from acting as a client in one version of NTP and a server in another.

3.6. Extensibility

To provide the protocol MUST have the capability to be extended. The specification should specify that implementations MUST ignore unknown extensions. Unknown extensions received by a server from a lower stratum server SHALL not be added to response messages sent by the server receiving these extensions.

4. IANA Considerations

Considerations should be made about the future of the existing IANA registry for NTPv4 parameters. If NTPv5 becomes incompatible with these parameters a new registry SHOULD be created.

5. Security Considerations

Encryption and authentication MUST be provided by the protocol specification as a default and MUST be resistent to downgrade attacks. The encryption used must have agility, allowing for the protocol to update as more secure cryptography becomes known and vulnerabilities are discovered.

The specification MAY consider leaving room for middleboxes which may deliberately modify packets in flight for legitimate purposes. Thought must be given around 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.

Detection and reporting of server malfeasence SHOULD remain out of scope of this specification as [I-D.ietf-ntp-roughtime] already provides this capability as a core functionality of the protocol.

--back

6. Acknowledgements

The author would like to thank Doug Arnold 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.

7. References

7.1. Normative References

[I-D.ietf-ntp-roughtime]
Malhotra, A., Langley, A., and W. Ladd, "Roughtime", draft-ietf-ntp-roughtime-03 (work in progress), August 2020.

[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>.

[RFC7808] Douglass, M. and C. Daboo, "Time Zone Data Distribution Service", RFC 7808, DOI 10.17487/RFC7808, March 2016, <https://www.rfc-editor.org/info/rfc7808>.

[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>.

7.2. Informative References

[IEEE-1588-2008]
"IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", n.d..

[ntppool] "pool.ntp.org: the internet cluster of ntp servers", n.d., <https://www.ntppool.org>.

7.3. URIs

[1] https://github.com/fiestajetsam/I-D/tree/main/draft-gruessing-ntp-ntpv5-requirements

Author's Address

James Gruessing

Email: james.ietf@gmail.com

Internet Engineering Task Force
Internet-Draft
Updates: 5905 (if approved)
Intended status: Standards Track
Expires: August 19, 2021

M. Lichvar
Red Hat

Feb 15, 2021

Alternative NTP port
draft-ietf-ntp-alternative-port-01

Abstract

This document updates RFC 5905 to specify an alternative port for the Network Time Protocol (NTP) which is restricted to NTP messages that do not allow traffic amplification.

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 August 19, 2021.

Table of Contents

1. Introduction

There are several modes specified for NTP. NTP packets in versions 2, 3, and 4 have a 3-bit field for the mode. Modes 1 (active), 2 (passive), 3 (client), 4 (server), and 5 (broadcast) are used for synchronization of clocks. They are specified in RFC 5905 [RFC5905]. Modes 6 and 7 are used for other purposes, like monitoring and remote management of NTP servers and clients. The mode 6 is specified in Control Messages Protocol for Use with Network Time Protocol Version 4 [I-D.ietf-ntp-mode-6-cmds].

The first group of modes typically does not allow any traffic amplification, i.e. the response is not larger than the request. An exception is Autokey [RFC5906], which allows an NTP response to be longer than the request, e.g. packets containing the Certificate Message or Cookie Message extension field. Autokey is rarely used. If it is enabled on a publicly accessible server, the access needs to be tightly controlled to limit denial-of-service (DoS) attacks exploiting the amplification.

The modes 6 and 7 of NTP allow significant traffic amplification, which has been exploited in large-scale DoS attacks on the Internet. Publicly accessible servers that support these modes need to be configured to not respond to requests using the modes, as recommended in BCP 233 [RFC8633], but the number of servers that still do that is significant enough to require specific mitigations.

Network operators have implemented different mitigations. They are not documented and may change over time. Some of the mitigations that have been observed are:

1. Blocked UDP packets with destination or source port 123

2. Blocked UDP packets with destination or source port 123 and specific length (e.g. longer than 48 octets)

3. Blocked UDP packets with destination or source port 123 and NTP mode 6 or 7

4. Limited rate of UDP packets with destination or source port 123

From those, only the 3rd approach does not have an impact on synchronization of clocks with NTP. However, this mitigation can be implemented only on devices which can inspect the UDP payload.

The number of public servers in the pool.ntp.org project has dropped since 2013, when the large-scale attacks started.

The length-specific filtering and rate limiting has an impact on the Network Time Security [RFC8915] authentication, which uses extension fields in NTPv4 packets.

This document specifies an alternative port for NTP which is restricted to a subset of the NTP protocol which does not allow amplification in order to enable safe synchronization of clocks in networks where the port 123 is blocked or rate limited.

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. Alternative port - update to RFC 5905

The table in "Figure 6: Global Parameters" in Section 7.2 of [RFC5905] is extended with:

Name	Value	Description
ALTPORT	TBD	Alternative NTP port

The following text from Section 9.1 of [RFC5905]:

> srcport: UDP port number of the server or reference clock. This becomes the destination port number in packets sent from this association. When operating in symmetric modes (1 and 2), this field must contain the NTP port number PORT (123) assigned by the IANA. In other modes, it can contain any number consistent with local policy.

is replaced with:

> srcport: UDP port number of the server or reference clock. This becomes the destination port number in packets sent from this association. When operating in symmetric modes (1 and 2), this field must contain the NTP port number PORT (123) or the alternative NTP port ALTPORT (TBD) assigned by the IANA. In other modes, it can contain any number consistent with local policy.

The following text is added to the Section 9.1:

> The port ALTPORT (TBD) is an alternative port to the port PORT (123). The protocol and the format of NTP packets sent from and to this port is unchanged. Both NTP requests and responses MAY be sent from the alternative port. An NTP packet MUST NOT be sent from the alternative port if it is a response which has a longer UDP payload than the request, or the number of NTP packets in a single response is larger than one.
>
> Only modes 1 (active), 2 (passive), 3 (client), 4 (server), and 5 (broadcast) are generally usable on this port.
>
> An NTP server that supports the alternative port MUST receive requests in the client mode on both the PORT (123) and ALTPORT (TBD) ports. If it responds, it MUST send the response from the port which received the request. If the server supports an NTP extension field, it MUST verify for each response that it is not longer than the request.
>
> When an NTP client is started, it SHOULD send the first request to the alternative port. The client SHOULD alternate between the two ports until a valid response is received. The client MAY send a limited number of requests to both ports at the same time in order to speed up the discovery of the responding port. When both ports are responding, the client SHOULD prefer the alternative port.
>
> An NTP server which supports NTS SHOULD include the NTPv4 Port Negotiation record in NTS-KE responses to specify the alternative port as the port to which the client should send NTP requests.
>
> In the symmetric modes (active and passive) NTP packets are considered to be requests and responses at the same time. Therefore, two peers using the alternative port MUST send packets with an equal length in order to synchronize with each other. The peers MAY still use different polling intervals as packets sent at subsequent polls are considered to be separate requests and responses.

3. IANA Considerations

IANA is requested to allocate the following port in the Service Name and Transport Protocol Port Number Registry [RFC6335]:

Service Name: ntp-alt

Transport Protocol: udp

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Network Time Protocol

Reference: [[this memo]]

Port Number: [[TBD]], selected by IANA from the System Port range

4. Security Considerations

A Man-in-the-middle (MITM) attacker can selectively block requests sent to the alternative port to force a client to select the original port and get a degraded NTP service with a significant packet loss. The client needs to periodically try the alternative port to recover from the degraded service when the attack stops.

5. Acknowledgements

The author would like to thank Daniel Franke, Dhruv Dhody, Ragnar Sundblad, and Steven Sommars for their useful comments.

6. References

6.1. Normative References

[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>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[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>.

6.2. Informative References

[I-D.ietf-ntp-mode-6-cmds] Haberman, B., "Control Messages Protocol for Use with Network Time Protocol Version 4", draft-ietf-ntp-mode-6-cmds-10 (work in progress), September 2020.

[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>.

[RFC8633] Reilly, D., Stenn, H., and D. Sibold, "Network Time Protocol Best Current Practices", BCP 223, RFC 8633, DOI 10.17487/RFC8633, July 2019, <https://www.rfc-editor.org/info/rfc8633>.

[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
Purkynova 115
Brno 612 00
Czech Republic

Email: mlichvar@redhat.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 25, 2021

N. Rozen-Schiff
D. Dolev
Hebrew University of Jerusalem
T. Mizrahi
Huawei Network.IO Innovation Lab
M. Schapira
Hebrew University of Jerusalem
February 21, 2021

A Secure Selection and Filtering Mechanism for the Network Time Protocol
draft-ietf-ntp-chronos-02

Abstract

The Network Time Protocol version 4 (NTPv4), as defined in RFC 5905, is the mechanism used by NTP clients to synchronize with NTP servers across the Internet. This document specifies an extension to the NTPv4 client, named Chronos, which is used as a "watchdog" alongside NTPv4, and provides improved security against time shifting attacks. Chronos involves changes to the NTP client's system process only and is backwards compatible with NTPv4 servers. Chronos is also compatible with NTPv5, since it does not affect the wire protocol.

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 August 25, 2021.

Copyright Notice

Table of Contents

1. Introduction

NTPv4, as defined in RFC 5905 [RFC5905], is vulnerable to time shifting attacks, in which the attacker's goal is to shift the local time at an NTP client. See [Chronos_paper] for details. Time shifting attacks on NTP are possible even if NTP communication is encrypted and authenticated. A weaker man-in-the-middle (MitM) attacker can shift time simply by dropping or delaying packets, whereas a powerful attacker, who has full control over an NTP server, can determine the response content. This document introduces a time shifting mitigation mechanism called Chronos. Chronos is backwards compatible with NTPv4 and serves as an NTPv4 client's "watchdog" for time shifting attacks. An NTP client that runs Chronos is interoperable with [RFC5905]-compatible NTPv4 servers. Chronos is also compatible with NTPv5, since it does not affect the wire protocol.

Chronos is a background mechanism that continuously maintains a virtual "Chronos" clock update and compares it to NTPv4's clock

update. When the gap between the two updates exceeds a certain threshold (specified in Section 6), this is interpreted as the client experiencing a time shifting attack. In this case, Chronos is used to update the client's clock, and NTPv4 is operated in the background until the gap between NTPv4 and Chronos' updates are again below this threshold, and hence NTPv4 is safe to use again.

Due to Chronos operating in the background, the client clock's precision and accuracy are precisely as in NTPv4 while not experiencing a time-shifting attack. When under attack, Chronos prevents the clock from being shifted by the attacker, thus still preserving high accuracy and precision (as discussed in Section 6).

Chronos achieves accurate synchronization even in the presence of powerful attackers who are in direct control of a large number of NTP servers: up to 1/3 of the servers in the pool (where the pool may consist of hundreds or even thousands of servers). NTPv4 chooses a small subset of the NTP server pool (e.g. 4 servers), and periodically queries this subset of servers. Thus, even if only 1/3 of the servers in the pool are compromised, the small subset that is used by NTPv4 may consist of a majority of faulty servers. Conversely, Chronos constantly updates the set of servers it queries; in each poll interval Chronos randomly chooses a different subset of servers from the pool. Thus, even if an attack is not detected in a given poll interval, Chronos is bound to detect the attack within a relatively small number of poll intervals.

A Chronos client iteratively "crowdsources" time queries across NTP servers and applies a provably secure algorithm for eliminating "suspicious" responses and for averaging over the remaining responses. Chronos is carefully engineered to minimize communication overhead so as to avoid overloading NTP servers. Chronos' security was evaluated both theoretically and experimentally with a prototype implementation. These evaluation results indicate that in order to successfully shift time at a Chronos client by over 100 milliseconds from the UTC, even a powerful man-in-the-middle attacker requires over 20 years of effort in expectation. The full paper is available at [Chronos_paper].

Chronos introduces a watchdog mechanism that is added to the client's system process and maintains a virtual clock value that is used as a reference for detecting attacks. The virtual clock value computation differs from the current NTPv4 in two key aspects. First, a Chronos client relies on a large number of NTP servers, from which only few servers to synchronize with are periodically chosen at random, in order to avoid overloading the servers. Second, the selection algorithm of the virtual clock uses an approximate agreement technique to remove outliers, thus limiting the attacker's ability to

contaminate the "time samples" (offsets) derived from the queried NTP servers. These two elements of Chronos' design provide provable security guarantees against both man-in-the-middle attackers and attackers capable of compromising a large number of NTP servers.

2. Conventions Used in This Document

2.1. Terminology

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].

2.2. Terms and Abbreviations

NTPv4	Network Time Protocol version 4 [RFC5905].
Selection process	Clock filter algorithm and system process [RFC5905].

2.3. Notations

Describing Chronos algorithm, the following notation are used.

Notation	Meaning
n	The number of candidate servers in the pool that Chronos can query (potentially hundreds)
m	The number of servers that NTPv4 queries in each poll interval (up to tens)
w	An upper bound on the distance of the local time from the UTC at any NTP server with an accurate clock (termed "truechimer" in [RFC5905])
Cest	The client's estimation for the time that has passed since its last synchronization to the server pool (sec)
B	An upper bound on the client's time estimation error (ms/sec)
ERR	An upper bound on the client's error regarding his estimation of the time passed from the last update, equals to B*Cest (ms)
K	Panic trigger - the number of pool re-sampling until reaches "Panic mode";
tc	The current time [sec], as indicated by the virtual clock value that is computed by Chronos

Table 1: Chronos Notations

The recommended values are discussed in Section 3.2.

3. Extension to the NTP System Process

A client that runs Chronos as a watchdog, uses NTPv4 as in [RFC5905] and in the background runs a modification to the elements of the system process described in Section 11.2.1 and 11.2.2 in [RFC5905] (namely, the Selection Algorithm and the Cluster Algorithm). The NTPv4 conventional protocol periodically queries m servers in each poll interval. In parallel the Chronos watchdog periodically queries a (variable) set of m servers in each Chronos poll interval. Specifically, in Chronos, after executing the "Clock Filter Algorithm" as defined in Section 10 in [RFC5905], the client discards outliers by executing the procedure described in this section and the next. Then, the NTPv4 "Combine Algorithm" is used for computing the system peer offset, as specified in Section 11.2.3 in [RFC5905]. In each poll interval the Chronos virtual clock value is compared with the NTPv4 clock value, and if the difference exceeds a predetermined value, an attack is detected. This process holds also for Chronos as a watchdog of future NTPv5.

3.1. Chronos' System Process

At the first time the Chronos system process is executed, calibration is needed. The calibration process generates a local pool of servers the client can synchronize with, consisting of n servers (up to hundreds). To this end, the NTP client executes the "Peer Process" and "Clock Filter Algorithm" as in Sections 9,10 in [RFC5905] (respectively), on an hourly basis, for 24 consecutive hours, and generates the union of all received NTP servers' IP addresses. Importantly, this process can also be executed in the background periodically, once in a long time (e.g., every few weeks/months).

In each Chronos poll interval the Chronos system process randomly chooses a set of m servers (where n with magnitude of hundreds and m of tens) out of the local pool of n servers. Then, out of the time-samples received from this chosen subset of servers, a lowest third of the samples' offset values and highest third of the samples' offset values are discarded.

Chronos checks that the following two conditions hold for the remaining samples:

- The maximal distance between every two time samples does not exceed 2w.

- The average value of the remaining samples is at distance at most ERR+2w from the client's local clock (as computed by Chronos).

(where w, ERR are as described in Table 1. Notice that ERR magnitude is approximately LAMBDA as defined in [RFC5905]).

In the event that both of these conditions are satisfied, the average of the remaining samples is the "final offset". Otherwise, a random partial of the interval is chosen, after which a new subset of servers is sampled, in the exact same manner. This way, Chronos client queries are spread across the time interval better in case of DoS attack on the NTP servers. This resampling process continues in subsequent Chronos poll intervals until the two conditions are both satisfied or the number of times the servers are re-sampled exceeds a "Panic Trigger" (K in Table 1), in which case, Chronos enters a "Panic Mode". Note that it is configurable whether the client allows panic mode or not.

In panic mode, Chronos queries all the servers in the local server pool, orders the collected time samples from lowest to highest and eliminates the bottom third and the top third of the samples. The client then averages over the remaining samples, and sets this average to be the new "final offset".

As in [RFC5905], the final offset is passed on to the clock discipline algorithm for the purpose of steering the Chronos virtual clock to the correct time. The Chronos virtual clock is then compared to the NTPv4 (or to the future NTPv5) clock as part of the watchdog process.

3.2. Chronos' Recommended Parameters

According to empirical observations (presented in [Chronos_paper]), querying 15 servers at each poll interval (i.e., m=15) out of 500 servers (n=500), and setting w to be around 25 milliseconds provides both high time accuracy and good security. Moreover, empirical analyses showed that, on average, approximately 83% of the servers' clocks are at most w-away from the UTC, and within 2w from each other, satisfying the first condition of Chronos' system process.

Furthermore, according to Chronos security analysis, setting K to be 3 (i.e., if after 3 re-sampling, the two conditions are not satisfied, then Chronos reaches "panic mode") is both safe when facing time shifting attacks and the probability of reaching the "panic mode" is negligible (less than 0.000002).

Chronos effect on precision and accuracy are discussed in Section 5 and Section 6.

4. Chronos' Pseudocode

The pseudocode for Chronos' Time Sampling Scheme, which is invoked in each Chronos poll interval is as follows:

```
counter := 0
S = []
T = []
While counter < K do
   S := sample(m) //gather samples from (tens of) randomly chosen servers
   T := bi-side-trim(S,1/3) //trim the third lowest and highest values
   if (max(T) -min(T) <= 2w) and (|avg(T)-tc| < ERR + 2w) Then
       return avg(t)
   end
   counter ++
   sleep(rand(0,1)*poll interval)
end
// panic mode
S := sample(n)
T := bi-sided-trim(S,1/3) //trim bottom and top thirds;
return avg(T)
```

5. Precision vs. Security

Since NTPv4 (and future NTPv5) updates the clock as long as time-shifting attacks are not detected, the precision and accuracy of a Chronos client are the same as NTPv4 when not under attack. Under attack, Chronos, which changes the list of the sampled servers more frequently than NTPv4 [Chronos_paper], and does not use some of the filters in NTPv4's system process, can potentially be less precise (though provably more secure than NTPv4, which is vulnerable to time-shifting attacks [RFC5905]).

However, our experimental and empirical analyses of Chronos revealed that Chornos and NTPv4 exhibit the same level of precision and accuracy when not under attack, with Chronos maintaining this level even in the presence of time-shifting attacks.

6. Chronos' Threat Model and Security Guarantees

As explained above, Chronos repeatedly gathers time samples from small subsets of a large local pool of NTP servers. The following form of a man-in-the-middle (MitM) Byzantine attacker is considered: the MitM attacker is assumed to control a subset of the servers in the local pool of servers and is capable of determining precisely the values of the time samples gathered by the Chronos client from these NTP servers. The threat model thus encompasses a broad spectrum of MitM attackers, ranging from fairly weak (yet dangerous) MitM attackers only capable of delaying and dropping packets to extremely powerful MitM attackers who are in control of (even authenticated) NTP servers. MitM attackers captured by this framework might be, for example, (1) in direct control of a fraction of the NTP servers (e.g., by exploiting a software vulnerability), (2) an ISP (or other Autonomous-System-level attacker) on the default BGP paths from the NTP client to a fraction of the available servers, (3) a nation state with authority over the owners of NTP servers in its jurisdiction, or (4) an attacker capable of hijacking (e.g., through DNS cache poisoning or BGP prefix hijacking) traffic to some of the available NTP servers. The details of the specific attack scenario are abstracted by reasoning about MitM attackers in terms of the fraction of servers with respect to which the attacker has MitM capabilities.

Chronos detects time-shifting attacks by constantly monitoring NTPv4's (or NTPv5's) offset and the offset computed by Chronos, as explained above, and checking whether it exceeds a certain threshold (10 milliseconds by default).

Analytical results (in [Chronos_paper]) indicate that in order to succeed in shifting time at a Chronos client by even a small amount (e.g., 100 milliseconds), even a powerful MitM attacker requires many

years of effort (e.g., over 20 years in expectation). See a brief overview of Chronos' security analysis below.

Notably, Chronos provides protection from MitM attacks that cannot be achieved by cryptographic authentication protocols since even with such measures in place an attacker can still influence time by dropping/delaying packets. However, adding an authentication and crypto-based security layer to Chronos will enhance its security guarantees and enable the detection of various spoofing and modification attacks.

Chronos' security analysis is briefly described next.

6.1. Security Analysis Overview

Time-samples that are at most w away from the UTC are considered "good", whereas other samples are considered "malicious". Two scenarios are considered:

- Less than 2/3 of the queried servers are under the attacker's control.
- The attacker controls more than 2/3 of the queried servers.

The first scenario, where there are more than 1/3 good samples, consists of two sub-cases: (i) there is at least one good sample in the set of samples not eliminated by Chronos (that is, in the middle third of samples), and (ii) there are no good samples in the remaining set of samples. In the first of these two cases (at least one good sample in the set of samples was not eliminated by Chronos), the other remaining samples, including those provided by the attacker, must be close to a good sample (for otherwise, the first condition of Chronos' system process in Section 3.1 is violated and a new set of servers is chosen). This implies that the average of the remaining samples must be close to the UTC. In the second case (there are no good samples in the set of remaining samples), since more than a third of the initial samples were good, both the (discarded) third lowest-value samples and the (discarded) third highest-value samples must each contain a good sample. Hence, all the remaining samples are bounded from both above and below by good samples, and so is their average value, implying that this value is close to the UTC [RFC5905].

In the second scenario, where the attacker controls more than 2/3 of the queried servers, the worst possibility for the client is that all remaining samples are malicious (i.e., more than w away from the UTC). However, as proved in [Chronos_paper], the probability of this scenario is extremely low even if the attacker controls a large

fraction (e.g., 1/4) of the servers in the local pool. The probability that the attacker repeatedly succeeds in realising this scenario decays exponentially, rendering the probability of a significant time shift negligible. See [Chronos_paper] for details.

Beyond evaluating the probability of an attacker successfully shifting time at the client's clock, we also evaluated the probability that the attacker succeeds in launching a DoS attack on the servers by causing many clients to enter panic mode (and so query all the servers in their local pools). This probability too is negligible even for an attacker in control of a large number of servers in clients' local server pools. See [Chronos_paper]for details.

Further details about Chronos's threat model and security guarantees can be found in [Chronos_paper].

7. Acknowledgements

The authors would like to thank Erik Kline, Miroslav Lichvar, Danny Mayer, Karen O'Donoghue, Dieter Sibold, Yaakov. J. Stein, and Harlan Stenn, for valuable contributions to this document and helpful discussions and comments.

8. IANA Considerations

This memo includes no request to IANA.

9. References

9.1. Normative References

[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>.

9.2. Informative References

[Chronos_paper]
Deutsch, O., Schiff, N., Dolev, D., and M. Schapira, "Preventing (Network) Time Travel with Chronos", 2018, <https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_02A-2_Deutsch_paper.pdf>.

Authors' Addresses

Neta Rozen-Schiff
Hebrew University of Jerusalem
Jerusalem
Israel

Phone: +972 2 549 4599
Email: neta.r.schiff@gmail.com

Danny Dolev
Hebrew University of Jerusalem
Jerusalem
Israel

Phone: +972 2 549 4588
Email: danny.dolev@mail.huji.ac.il

Tal Mizrahi
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@gmail.com

Michael Schapira
Hebrew University of Jerusalem
Jerusalem
Israel

Phone: +972 2 549 4570
Email: schapiram@huji.ac.il

Internet Engineering Task Force
Internet-Draft
Updates: 5905 (if approved)
Intended status: Standards Track
Expires: October 10, 2021

M. Lichvar
Red Hat
A. Malhotra
Boston University

Apr 8, 2021

NTP Interleaved Modes
draft-ietf-ntp-interleaved-modes-05

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 October 10, 2021.

Copyright Notice

Table of Contents

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 an 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 single packet, which contains a transmit timestamp corresponding to the previous packet that was sent to the client or peer. This transmit timestamp can be captured at any of the four places mentioned above. Both servers and clients/peers are required to keep some state specific to the interleaved mode.

The protocol does not change the NTP packet header format, only the semantics of some timestamp fields. 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 negotiation in the protocol is implicit. The origin timestamp enables servers and peers 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,

which would not work well with implementations that do not respond to requests with unknown extension fields.

Requests and responses cannot always be formed in interleaved mode. Servers, clients, and peers are required to support both interleaved and basic modes.

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 only difference between the two modes is in the meaning of the transmit and origin timestamp fields.

The origin timestamp is a cookie, which is used to detect a packet which is not a response to the last packet sent in the other direction. Such packets are called bogus packets in RFC 5905.

A client request in the basic mode has an origin timestamp equal to the transmit timestamp from the previous server response, or is zero. A server response in the basic mode has an origin timestamp equal to the transmit timestamp from the client's request. The transmit timestamps correspond to the packets in which they are included.

A client request in the interleaved mode has an origin timestamp equal to the receive timestamp from the previous server response. A server response in the interleaved mode has an origin timestamp equal to the receive timestamp from the client's request. The transmit timestamps correspond to the previous packets that were sent to the server or client.

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, i.e. clients are allowed to change their source port between requests.

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.

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 timestamp is 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~, t3~ and t11~ 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).

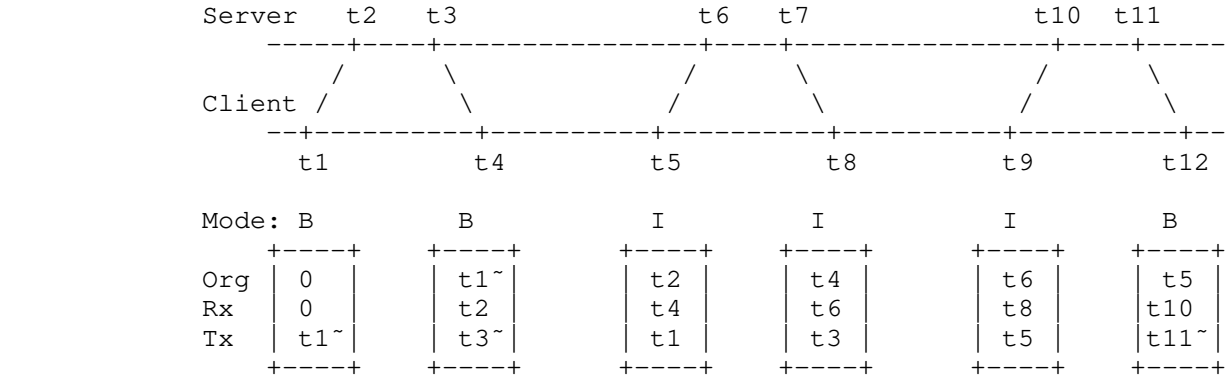


Figure 1: Packet timestamps in interleaved client/server mode

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 clients that implement NTP data minimization [I-D.ietf-ntp-data-minimization]. To detect requests in the basic mode from clients that do not implement the data minimization, the server can encode in low-order bits of the receive and transmit timestamps below precision of the clock a bit 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 is in the basic mode and it is not necessary to 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 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 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.

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 are in the basic mode, because multiple responses are sent per request.

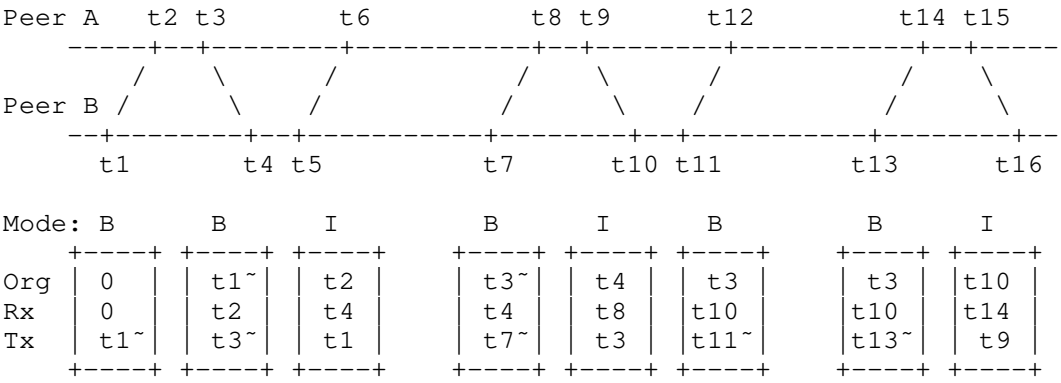


Figure 2: Packet timestamps in interleaved symmetric mode

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.

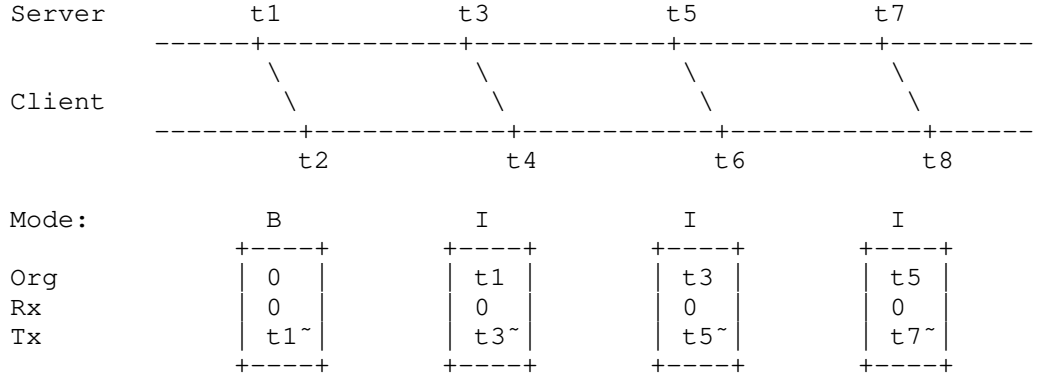


Figure 3: Packet timestamps in interleaved broadcast mode

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.

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. Acknowledgements

The interleaved modes described in this document are based on the implementation written by David Mills in the NTP project [1]. 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, Erik Kline, Tal Mizrahi, Steven Sommars, Harlan Stenn, and Kristof Teichel for their useful comments.

6. IANA Considerations

This memo includes no request to IANA.

7. 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. It is not possible to zero the origin timestamp to prevent passive observers from easily tracking clients moving between different 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. The NTP state needs to be protected not only between the reception and transmission in order to send the peer a packet with a valid origin timestamp, but all the time to not lose the timestamps which will be

needed to complete a measurement when the following packet in the interleaved mode is received.

8. References

8.1. Normative References

[I-D.ietf-ntp-data-minimization]
Franke, D. and A. Malhotra, "NTP Client Data Minimization", draft-ietf-ntp-data-minimization-04 (work in progress), March 2019.

[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>.

8.2. Informative References

[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>.

[SECNTP] Malhotra, A., Gundy, M., Varia, M., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", 2016, <http://eprint.iacr.org/2016/1006>.

8.3. URIs

[1] http://www.ntp.org

Authors' Addresses

Miroslav Lichvar
Red Hat
Purkynova 115
Brno 612 00
Czech Republic

Email: mlichvar@redhat.com

Aanchal Malhotra
Boston University
111 Cummington St
Boston 02215
USA

Email: aanchal4@bu.edu

Network Working Group
Internet-Draft
Intended status: Historic
Expires: April 1, 2021

B. Haberman, Ed.
JHU

September 28, 2020

Control Messages Protocol for Use with Network Time Protocol Version 4
draft-ietf-ntp-mode-6-cmds-10

Abstract

This document describes the structure of the control messages that were historically used with the Network Time Protocol before the advent of more modern control and management approaches. These control messages have been used to monitor and control the Network Time Protocol application running on any IP network attached computer. The information in this document was originally described in Appendix B of RFC 1305. The goal of this document is to provide an updated description of the control messages described in RFC 1305 in order to conform with the updated Network Time Protocol specification documented in RFC 5905.

The publication of this document is not meant to encourage the development and deployment of these control messages. This document is only providing a current reference for these control messages given the current status of RFC 1305.

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 April 1, 2021.

Table of Contents

1. Introduction

RFC 1305 [RFC1305] described a set of control messages for use within the Network Time Protocol (NTP) when a comprehensive network management solution was not available. The definitions of these control messages were not promulgated to RFC 5905 [RFC5905] when NTP version 4 was documented. These messages were intended for use only in systems where no other management facilities were available or appropriate, such as in dedicated-function bus peripherals. Support for these messages is not required in order to conform to RFC 5905 [RFC5905]. The control messages are described here as a current reference for use with an RFC 5905 implementation of NTP.

The publication of this document is not meant to encourage the development and deployment of these control messages. This document is only providing a current reference for these control messages given the current status of RFC 1305.

1.1. Control Message Overview

The NTP Mode 6 control messages are used by NTP management programs (e.g., ntpq) when a more robust network management facility (e.g., SNMP) is not available. These control messages provide rudimentary control and monitoring functions to manage a running instance of an NTP server. These commands are not designed to be used for communication between instances of running NTP servers.

The NTP Control Message has the value 6 specified in the mode field of the first octet of the NTP header and is formatted as shown in Figure 1. The format of the data field is specific to each command or response; however, in most cases the format is designed to be constructed and viewed by humans and so is coded in free-form ASCII. This facilitates the specification and implementation of simple management tools in the absence of fully evolved network-management facilities. As in ordinary NTP messages, the authenticator field follows the data field. If the authenticator is used the data field is zero-padded to a 32-bit boundary, but the padding bits are not considered part of the data field and are not included in the field count.

IP hosts are not required to reassemble datagrams over a certain size (576 octets for IPv4 [RFC0791] and 1280 octets for IPv6 [RFC2460]); however, some commands or responses may involve more data than will fit into a single datagram. Accordingly, a simple reassembly feature is included in which each octet of the message data is numbered starting with zero. As each fragment is transmitted the number of its first octet is inserted in the offset field and the number of

octets is inserted in the count field. The more-data (M) bit is set in all fragments except the last.

Most control functions involve sending a command and receiving a response, perhaps involving several fragments. The sender chooses a distinct, nonzero sequence number and sets the status field and "R" and "E" bits to zero. The responder interprets the opcode and additional information in the data field, updates the status field, sets the "R" bit to one and returns the three 32-bit words of the header along with additional information in the data field. In case of invalid message format or contents the responder inserts a code in the status field, sets the "R" and "E" bits to one and, optionally, inserts a diagnostic message in the data field.

Some commands read or write system variables (e.g., s.offset) and peer variables (e.g., p.stratum) for an association identified in the command. Others read or write variables associated with a radio clock or other device directly connected to a source of primary synchronization information. To identify which type of variable and association the Association ID is used. System variables are indicated by the identifier zero. As each association is mobilized a unique, nonzero identifier is created for it. These identifiers are used in a cyclic fashion, so that the chance of using an old identifier which matches a newly created association is remote. A management entity can request a list of current identifiers and subsequently use them to read and write variables for each association. An attempt to use an expired identifier results in an exception response, following which the list can be requested again.

Some exception events, such as when a peer becomes reachable or unreachable, occur spontaneously and are not necessarily associated with a command. An implementation may elect to save the event information for later retrieval or to send an asynchronous response (called a trap) or both. In case of a trap the IP address and port number is determined by a previous command and the sequence field is set as described below. Current status and summary information for the latest exception event is returned in all normal responses. Bits in the status field indicate whether an exception has occurred since the last response and whether more than one exception has occurred.

Commands need not necessarily be sent by an NTP peer, so ordinary access-control procedures may not apply; however, the optional mask/match mechanism suggested in Section Section 6 elsewhere in this document provides the capability to control access by mode number, so this could be used to limit access for control messages (mode 6) to selected address ranges.

1.2. Remote Facility Message Overview

The original development of the NTP daemon included a remote facility for monitoring and configuration. This facility used mode 7 commands to communicate with the NTP daemon. This document illustrates the mode 7 packet format only. The commands embedded in the mode 7 messages are implementation specific and not standardized in any way. The mode 7 message format is described in Appendix A.

2. NTP Control Message Format

The format of the NTP Control Message header, which immediately follows the UDP header, is shown in Figure 1. Following is a description of its fields.

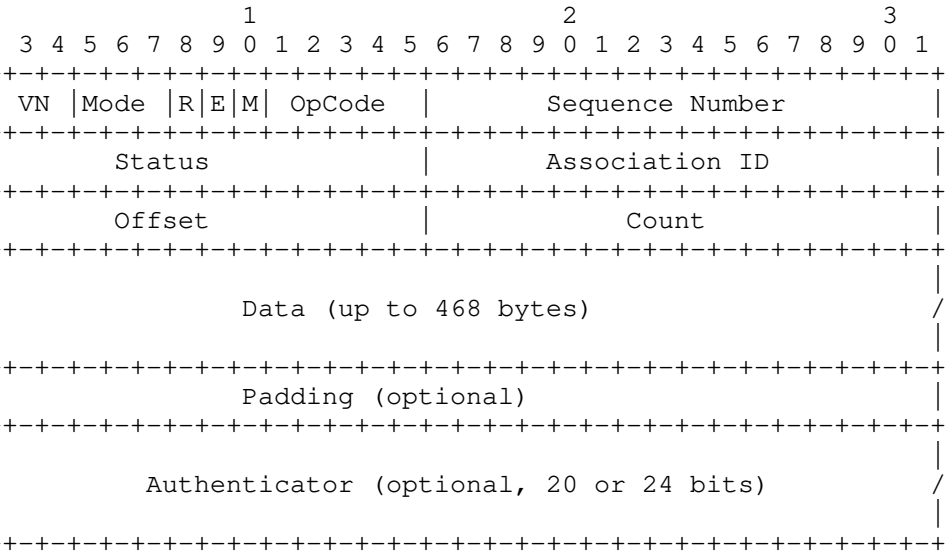


Figure 1: NTP Control Message Header

Leap Indicator (LI): This is a two-bit integer that is set to b00 for control message requests and responses. The Leap Indicator value used at this position in most NTP modes is in the System Status Word provided in some control message responses.

Version Number (VN): This is a three-bit integer indicating a minimum NTP version number. NTP servers do not respond to control messages with an unrecognized version number. Requests may intentionally use a lower version number to enable interoperability with earlier versions of NTP. Responses carry the same version as the corresponding request.

Mode: This is a three-bit integer indicating the mode. The value 6 indicates an NTP control message.

Response Bit (R): Set to zero for commands, one for responses.

Error Bit (E): Set to zero for normal response, one for error response.

More Bit (M): Set to zero for last fragment, one for all others.

Operation Code (OpCode): This is a five-bit integer specifying the command function. Values currently defined include the following:

Code	Meaning
0	reserved
1	read status command/response
2	read variables command/response
3	write variables command/response
4	read clock variables command/response
5	write clock variables command/response
6	set trap address/port command/response
7	trap response
8	runtime configuration command/response
9	export configuration to file command/response
10	retrieve remote address stats command/response
11	retrieve ordered list command/response
12	request client-specific nonce command/response
13-30	reserved
31	unset trap address/port command/response

Sequence Number: This is a 16-bit integer indicating the sequence number of the command or response. Each request uses a different sequence number. Each response carries the same sequence number as its corresponding request. For asynchronous trap responses, the responder increments the sequence number by one for each response, allowing trap receivers to detect missing trap responses. The sequence number of each fragment of a multiple-datagram response carries the same sequence number, copied from the request.

Status: This is a 16-bit code indicating the current status of the system, peer or clock, with values coded as described in following sections.

Association ID: This is a 16-bit unsigned integer identifying a valid association, or zero for the system clock.

Offset: This is a 16-bit unsigned integer indicating the offset, in octets, of the first octet in the data area. The offset is set to zero in requests. Responses spanning multiple datagrams use a positive offset in all but the first datagram.

Count: This is a 16-bit unsigned integer indicating the length of the data field, in octets.

Data: This contains the message data for the command or response. The maximum number of data octets is 468.

Padding (optional): Contains zero to three octets with value zero, as needed to ensure the overall control message size is a multiple of 4 octets.

Authenticator (optional): When the NTP authentication mechanism is implemented, this contains the authenticator information defined in Appendix C of [RFC1305].

3. Status Words

Status words indicate the present status of the system, associations and clock. They are designed to be interpreted by network-monitoring programs and are in one of four 16-bit formats shown in Figure 2 and described in this section. System and peer status words are associated with responses for all commands except the read clock variables, write clock variables and set trap address/port commands. The association identifier zero specifies the system status word, while a nonzero identifier specifies a particular peer association. The status word returned in response to read clock variables and write clock variables commands indicates the state of the clock hardware and decoding software. A special error status word is used to report malformed command fields or invalid values.

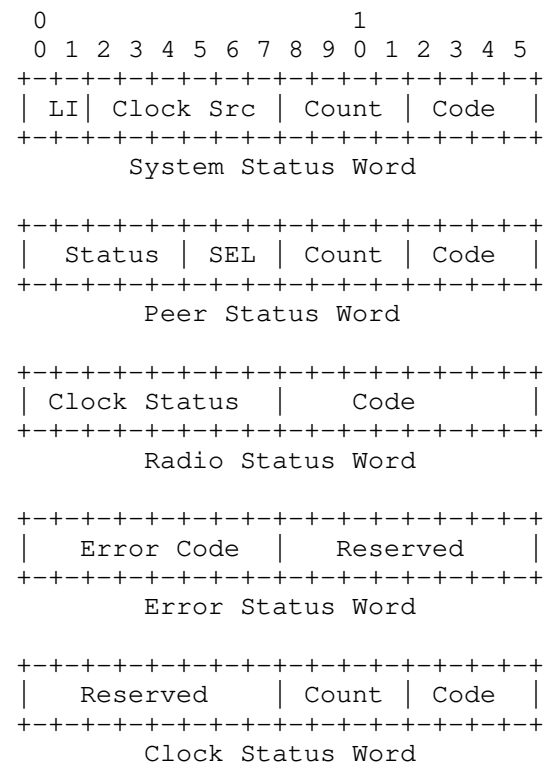


Figure 2: Status Word Formats

3.1. System Status Word

The system status word appears in the status field of the response to a read status or read variables command with a zero association identifier. The format of the system status word is as follows:

Leap Indicator (LI): This is a two-bit code warning of an impending leap second to be inserted/deleted in the last minute of the current day, with bit 0 and bit 1, respectively, coded as follows:

LI	Meaning
00	no warning
01	insert second after 23:59:59 of the current day
10	delete second 23:59:59 of the current day
11	unsynchronized

Clock Source (Clock Src): This is a six-bit integer indicating the current synchronization source, with values coded as follows:

Code	Meaning
0	unspecified or unknown
1	Calibrated atomic clock (e.g., PPS, HP 5061)
2	VLF (band 4) or LF (band 5) radio (e.g., OMEGA,, WWVB)
3	HF (band 7) radio (e.g., CHU, MSF, WWV/H)
4	UHF (band 9) satellite (e.g., GOES, GPS)
5	local net (e.g., DCN, TSP, DTS)
6	UDP/NTP
7	UDP/TIME
8	eyeball-and-wristwatch
9	telephone modem (e.g., NIST)
10-63	reserved

System Event Counter (Count): This is a four-bit integer indicating the number of system events occurring since the last time the System Event Code changed. Upon reaching 15, subsequent events with the same code are not counted.

System Event Code (Code): This is a four-bit integer identifying the latest system exception event, with new values overwriting previous values, and coded as follows:

Code	Meaning
0	unspecified
1	frequency correction (drift) file not available
2	frequency correction started (frequency stepped)
3	spike detected and ignored, starting stepout timer
4	frequency training started
5	clock synchronized
6	system restart
7	panic stop (required step greater than panic threshold)
8	no system peer
9	leap second insertion/deletion armed for the of the current month
10	leap second disarmed
11	leap second inserted or deleted
12	clock stepped (stepout timer expired)
13	kernel loop discipline status changed
14	leapseconds table loaded from file
15	leapseconds table outdated, updated file needed

3.2. Peer Status Word

A peer status word is returned in the status field of a response to a read status, read variables or write variables command and appears also in the list of association identifiers and status words returned by a read status command with a zero association identifier. The format of a peer status word is as follows:

Peer Status (Status): This is a five-bit code indicating the status of the peer determined by the packet procedure, with bits assigned as follows:

Peer Status bit	Meaning
0	configured (peer.config)
1	authentication enabled (peer.authenable)
2	authentication okay (peer.authentic)
3	reachability okay (peer.reach != 0)
4	broadcast association

Peer Selection (SEL): This is a three-bit integer indicating the status of the peer determined by the clock-selection procedure, with values coded as follows:

Sel	Meaning
0	rejected
1	discarded by intersection algorithm
2	discarded by table overflow (not currently used)
3	discarded by the cluster algorithm
4	included by the combine algorithm
5	backup source (with more than sys.maxclock survivors)
6	system peer (synchronization source)
7	PPS (pulse per second) peer

Peer Event Counter (Count): This is a four-bit integer indicating the number of peer exception events that occurred since the last time the peer event code changed. Upon reaching 15, subsequent events with the same code are not counted.

Peer Event Code (Code): This is a four-bit integer identifying the latest peer exception event, with new values overwriting previous values, and coded as follows:

Peer Event Code	Meaning
0	unspecified
1	association mobilized
2	association demobilized
3	peer unreachable (peer.reach was nonzero now zero)
4	peer reachable (peer.reach was zero now nonzero)
5	association restarted or timed out
6	no reply (only used with one-shot clock set command)
7	peer rate limit exceeded (kiss code RATE received)
8	access denied (kiss code DENY received)
9	leap second insertion/deletion at month's end armed by peer vote
10	became system peer (sys.peer)
11	reference clock event (see clock status word)
12	authentication failed
13	popcorn spike suppressed by peer clock filter register
14	entering interleaved mode
15	recovered from interleave error

3.3. Clock Status Word

There are two ways a reference clock can be attached to a NTP service host, as a dedicated device managed by the operating system and as a synthetic peer managed by NTP. As in the read status command, the association identifier is used to identify which one, zero for the system clock and nonzero for a peer clock. Only one system clock is supported by the protocol, although many peer clocks can be supported. A system or peer clock status word appears in the status field of the response to a read clock variables or write clock variables command. This word can be considered an extension of the system status word or the peer status word as appropriate. The format of the clock status word is as follows:

Reserved: An eight-bit integer that is ignored by requesters and zeroed by responders.

Count: This is a four-bit integer indicating the number of clock events that occurred since the last time the clock event code changed. Upon reaching 15, subsequent events with the same code are not counted.

Clock Code (Code): This is a four-bit integer indicating the current clock status, with values coded as follows:

Clock Status	Meaning
0	clock operating within nominals
1	reply timeout
2	bad reply format
3	hardware or software fault
4	propagation failure
5	bad date format or value
6	bad time format or value
7-15	reserved

3.4. Error Status Word

An error status word is returned in the status field of an error response as the result of invalid message format or contents. Its presence is indicated when the E (error) bit is set along with the response (R) bit in the response. It consists of an eight-bit integer coded as follows:

Error Status	Meaning
0	unspecified
1	authentication failure
2	invalid message length or format
3	invalid opcode
4	unknown association identifier
5	unknown variable name
6	invalid variable value
7	administratively prohibited
8-255	reserved

4. Commands

Commands consist of the header and optional data field shown in Figure 1. When present, the data field contains a list of identifiers or assignments in the form <<identifier>>[=<<value>>],<<identifier>>[=<<value>>],... where <<identifier>> is the ASCII name of a system or peer variable such as the ones specified in RFC 5905 and <<value>> is expressed as a decimal, hexadecimal or string constant in the syntax of the C programming language. Where no ambiguity exists, the "sys." or "peer." prefixes can be suppressed. Whitespace (ASCII nonprinting format effectors) can be added to improve readability for simple monitoring programs that do not reformat the data field. Internet addresses are represented as follows: IPv4 addresses are written in the form [n.n.n.n], where n is in decimal notation and the brackets are optional; IPv6 addresses are formulated based on the guidelines defined in [RFC5952]. Timestamps, including reference, originate, receive and transmit values, as well as the logical clock, are represented in units of seconds and fractions, preferably in hexadecimal notation. Delay, offset, dispersion and distance values are represented in units of milliseconds and fractions, preferably in decimal notation. All other values are represented as-is, preferably in decimal notation.

Implementations may define variables other than those described in RFC 5905. Called extramural variables, these are distinguished by the inclusion of some character type other than alphanumeric or "." in the name. For those commands that return a list of assignments in the response data field, if the command data field is empty, it is expected that all available variables defined in RFC 5905 will be included in the response. For the read commands, if the command data field is nonempty, an implementation may choose to process this field to individually select which variables are to be returned.

Commands are interpreted as follows:

Read Status (1): The command data field is empty or contains a list of identifiers separated by commas. The command operates in two ways depending on the value of the association identifier. If this identifier is nonzero, the response includes the peer identifier and status word. Optionally, the response data field may contain other information, such as described in the Read Variables command. If the association identifier is zero, the response includes the system identifier (0) and status word, while the data field contains a list of binary-coded pairs <<association identifier>> <<status word>>, one for each currently defined association.

Read Variables (2): The command data field is empty or contains a list of identifiers separated by commas. If the association identifier is nonzero, the response includes the requested peer identifier and status word, while the data field contains a list of peer variables and values as described above. If the association identifier is zero, the data field contains a list of system variables. If a peer has been selected as the synchronization source, the response includes the peer identifier and status word; otherwise, the response includes the system identifier (0) and status word.

Write Variables (3): The command data field contains a list of assignments as described above. The variables are updated as indicated. The response is as described for the Read Variables command.

Read Clock Variables (4): The command data field is empty or contains a list of identifiers separated by commas. The association identifier selects the system clock variables or peer clock variables in the same way as in the Read Variables command. The response includes the requested clock identifier and status word and the data field contains a list of clock variables and values, including the last timecode message received from the clock.

Write Clock Variables (5): The command data field contains a list of assignments as described above. The clock variables are updated as indicated. The response is as described for the Read Clock Variables command.

Set Trap Address/Port (6): The command association identifier, status and data fields are ignored. The address and port number for subsequent trap messages are taken from the source address and port of the control message itself. The initial trap counter for trap response messages is taken from the sequence field of the command. The response association identifier, status and data fields are not

significant. Implementations should include sanity timeouts which prevent trap transmissions if the monitoring program does not renew this information after a lengthy interval.

Trap Response (7): This message is sent when a system, peer or clock exception event occurs. The opcode field is 7 and the R bit is set. The trap counter is incremented by one for each trap sent and the sequence field set to that value. The trap message is sent using the IP address and port fields established by the set trap address/port command. If a system trap the association identifier field is set to zero and the status field contains the system status word. If a peer trap the association identifier field is set to that peer and the status field contains the peer status word. Optional ASCII-coded information can be included in the data field.

Configure (8): The command data is parsed and applied as if supplied in the daemon configuration file.

Save Configuration (9): Write a snapshot of the current configuration to the file name supplied as the command data. Further, the command is refused unless a directory in which to store the resulting files has been explicitly configured by the operator.

Read Most Recently Used (MRU) list (10): Retrieves records of recently seen remote addresses and associated statistics. Command data consists of name=value pairs controlling the selection of records, as well as a requestor-specific nonce previously retrieved using this command or opcode 12, Request Nonce. The response consists of name=value pairs where some names can appear multiple times using a dot followed by a zero-based index to distinguish them, and to associate elements of the same record with the same index. A new nonce is provided with each successful response.

Read ordered list (11): Retrieves a list ordered by IP address (IPv4 information precedes IPv6 information). If the command data is empty or the seven characters "ifstats", the associated statistics, status and counters for each local address are returned. If the command data is the characters "addr_restrictions" then the set of IPv4 remote address restrictions followed by the set of IPv6 remote address restrictions (access control lists) are returned. Other command data returns error code 5 (unknown variable name). Similar to Read MRU, response information uses zero-based indexes as part of the variable name preceding the equals sign and value, where each index relates information for a single address or network. This opcode requires authentication.

Request Nonce (12): Retrieves a 96-bit nonce specific to the requesting remote address, which is valid for a limited period.

Command data is not used in the request. The nonce consists of a 64-bit NTP timestamp and 32 bits of hash derived from that timestamp, the remote address, and salt known only to the server which varies between daemon runs. Inclusion of the nonce by a management agent demonstrates to the server that the agent can receive datagrams sent to the source address of the request, making source address "spoofing" more difficult in a similar way as TCP's three-way handshake.

Unset Trap (31): Removes the requesting remote address and port from the list of trap receivers. Command data is not used in the request. If the address and port are not in the list of trap receivers, the error code is 4, bad association.

5. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

A number of security vulnerabilities have been identified with these control messages.

NTP's control query interface allows reading and writing of system, peer, and clock variables remotely from arbitrary IP addresses using commands mentioned in Section 4. Traditionally, overwriting these variables, but not reading them, requires authentication by default. However, this document argues that an NTP host must authenticate all control queries and not just ones that overwrite these variables. Alternatively, the host can use an access control list to explicitly list IP addresses that are allowed to control query the clients. These access controls are required for the following reasons:

- NTP as a Distributed Denial-of-Service (DDoS) vector. NTP timing query and response packets (modes 1-2, 3-4, 5) are usually short in size. However, some NTP control queries generate a very long packet in response to a short query. As such, there is a history of use of NTP's control queries, which exhibit such behavior, to perform DDoS attacks. These off-path attacks exploit the large size of NTP control queries to cause UDP-based amplification attacks (e.g., mode 7 monlist command generates a very long packet in response to a small query [CVE-DOS]). These attacks only use NTP as a vector for DoS attacks on other protocols, but do not affect the time service on the NTP host itself. To limit the

sources of these malicious commands, NTP server operators are recommended to deploy ingress filtering [RFC3704].

- Time-shifting attacks through information leakage/overwriting. NTP hosts save important system and peer state variables. An off-path attacker who can read these variables remotely can leverage the information leaked by these control queries to perform time-shifting and DoS attacks on NTP clients. These attacks do affect time synchronization on the NTP hosts. For instance,

  * In the client/server mode, the client stores its local time when it sends the query to the server in its xmt peer variable. This variable is used to perform TEST2 to non-cryptographically authenticate the server, i.e., if the origin timestamp field in the corresponding server response packet matches the xmt peer variable, then the client accepts the packet. An off-path attacker, with the ability to read this variable can easily spoof server response packets for the client, which will pass TEST2, and can deny service or shift time on the NTP client. The specific attack is described in [CVE-SPOOF].

  * The client also stores its local time when the server response is received in its rec peer variable. This variable is used for authentication in interleaved-pivot mode. An off-path attacker with the ability to read this state variable can easily shift time on the client by passing this test. This attack is described in [CVE-SHIFT].

- Fast-Scanning. NTP mode 6 control messages are usually small UDP packets. Fast-scanning tools like ZMap can be used to spray the entire (potentially reachable) Internet with these messages within hours to identify vulnerable hosts. To make things worse, these attacks can be extremely low-rate, only requiring a control query for reconnaissance and a spoofed response to shift time on vulnerable clients.

- The mode 6 and 7 messages are vulnerable to replay attacks [CVE-Replay]. If an attacker observes mode 6/7 packets that modify the configuration of the server in any way, the attacker can apply the same change at any time later simply by sending the packets to the server again. The use of the nonce (Request Nonce command) provides limited protection against replay attacks.

NTP best practices recommend configuring NTP with the no-query parameter. The no-query parameter blocks access to all remote control queries. However, sometimes the hosts do not want to block all queries and want to give access for certain control queries remotely. This could be for the purpose of remote management and

configuration of the hosts in certain scenarios. Such hosts tend to use firewalls or other middleboxes to blacklist certain queries within the network.

Significantly fewer hosts respond to mode 7 monlist queries as compared to other control queries because it is a well-known and exploited control query. These queries are likely blocked using blacklists on firewalls and middleboxes rather than the no-query option on NTP hosts. The remaining control queries that can be exploited likely remain out of the blacklist because they are undocumented in the current NTP specification [RFC5905].

This document describes all of the mode 6 control queries allowed by NTP and can help administrators make informed decisions on security measures to protect NTP devices from harmful queries and likely make those systems less vulnerable. Regardless of which mode 6 commands an administrator may elect to allow, remote access to this facility needs to be protected from unauthorized access (e.g., strict ACLs).

7. Contributors

Dr. David Mills specified the vast majority of the mode 6 commands during the development of RFC 1305 [RFC1305] and deserves the credit for their existence and use.

8. Acknowledgements

Tim Plunkett created the original version of this document. Aanchal Malhotra provided the initial version of the Security Considerations section.

Karen O'Donoghue, David Hart, Harlan Stenn, and Philip Chimento deserve credit for portions of this document due to their earlier efforts to document these commands.

Miroshav Lichvar, Ulrich Windl, Dieter Sibold, J Ignacio Alvarez-Hamelin, and Alex Campbell provided valuable comments on various versions of this document.

9. References

9.1. Normative References

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, DOI 10.17487/RFC1305, March 1992, <https://www.rfc-editor.org/info/rfc1305>.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <https://www.rfc-editor.org/info/rfc3704>.

[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>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

9.2. Informative References

[CVE-DOS] NIST National Vulnerability Database, "CVE-2013-5211, https://nvd.nist.gov/vuln/detail/CVE-2013-5211", January 2014.

[CVE-Replay] NIST National Vulnerability Database, "CVE-2015-8140, https://nvd.nist.gov/vuln/detail/CVE-2015-8140", January 2015.

[CVE-SHIFT] NIST National Vulnerability Database, "CVE-2016-1548, https://nvd.nist.gov/vuln/detail/CVE-2016-1548", January 2017.

[CVE-SPOOF] NIST National Vulnerability Database, "CVE-2015-8139, https://nvd.nist.gov/vuln/detail/CVE-2015-8139", January 2017.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

Appendix A. NTP Remote Facility Message Format

The format of the NTP Remote Facility Message header, which immediately follows the UDP header, is shown in Figure 3. Following

is a description of its fields. Bit positions marked as zero are reserved and should always be transmitted as zero.

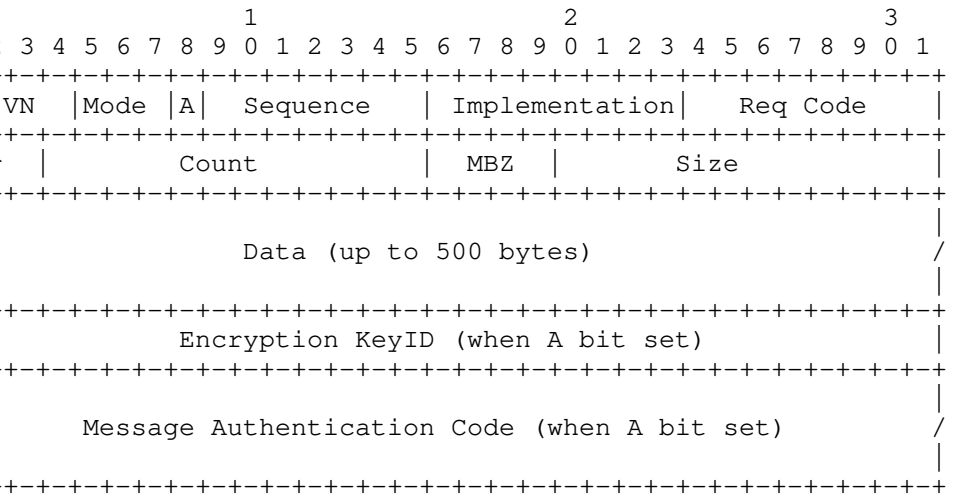


Figure 3: NTP Remote Facility Message Header

Response Bit (R) : Set to 0 if the packet is a request. Set to 1 if the packet is a response.

More Bit (M) : Set to 0 if this is the last packet in a response, otherwise set to 1 in responses requiring more than one packet.

Version Number (VN) : Set to the version number of the NTP daemon.

Mode : Set to 7 for Remote Facility messages.

Authenticated Bit (A) : If set to 1, this packet contains authentication information.

Sequence : For a multi-packet response, this field contains the sequence number of this packet. Packets in a multi-packet response are numbered starting with 0. The More Bit is set to 1 for all packets but the last.

Implementation : The version number of the implementation that defined the request code used in this message. An implementation number of 0 is used for a Request Code supported by all versions of the NTP daemon. The value 255 is reserved for future extensions.

Request Code (Req Code) : An implementation-specific code which specifies the operation being requested. A Request Code definition includes the format and semantics of the data included in the packet.

Error (Err) : Set to 0 for a request. For a response, this field contains an error code relating to the request. If the Error is non-zero, the operation requested wasn't performed.

0 - no error

1 - incompatible implementation number

2 - unimplemented request code

3 - format error

4 - no data available

7 - authentication failure

Count : The number of data items in the packet. Range is 0 to 500.

Must Be Zero (MBZ) : A reserved field set to 0 in requests and responses.

Size : The size of each data item in the packet. Range is 0 to 500.

Data : A variable-sized field containing request/response data. For requests and responses, the size in octets must be greater than or equal to the product of the number of data items (Count) and the size of a data item (Size). For requests, the data area is exactly 40 octets in length. For responses, the data area will range from 0 to 500 octets, inclusive.

Encryption KeyID : A 32-bit unsigned integer used to designate the key used for the Message Authentication Code. This field is included only when the A bit is set to 1.

Message Authentication Code : An optional Message Authentication Code defined by the version of the NTP daemon indicated in the Implementation field. This field is included only when the A bit is set to 1.

Author's Address

Brian Haberman (editor)
JHU

Email: brian@innovationslab.net

Network Time Protocol (ntp) Working Group
Internet-Draft
Updates: 5905 (if approved)
Intended status: Standards Track
Expires: March 19, 2021

F. Gont
G. Gont
SI6 Networks
M. Lichvar
Red Hat

September 15, 2020

Port Randomization in the Network Time Protocol Version 4
draft-ietf-ntp-port-randomization-06

Abstract

The Network Time Protocol can operate in several modes. Some of these modes are based on the receipt of unsolicited packets, and therefore require the use of a well-known port as the local port number. However, in the case of NTP modes where the use of a well-known port is not required, employing such well-known port unnecessarily increases the ability of attackers to perform blind/off-path attacks. This document formally updates RFC5905, recommending the use of transport-protocol ephemeral port randomization for those modes where use of the NTP well-known port is not required.

Table of Contents

1. Introduction

The Network Time Protocol (NTP) is one of the oldest Internet protocols, and currently specified in [RFC5905]. Since its original implementation, standardization, and deployment, a number of vulnerabilities have been found both in the NTP specification and in some of its implementations [NTP-VULN]. Some of these vulnerabilities allow for off-path/blind attacks, where an attacker can send forged packets to one or both NTP peers for achieving Denial of Service (DoS), time-shifts, or other undesirable outcomes. Many of these attacks require the attacker to guess or know at least a target NTP association, typically identified by the tuple {srcaddr, srcport, dstaddr, dstport, keyid} (see section 9.1 of [RFC5905]). Some of these parameters may be easily known or guessed.

NTP can operate in several modes. Some of these modes rely on the ability of nodes to receive unsolicited packets, and therefore require the use of the NTP well-known port (123). However, for modes where the use of a well-known port is not required, employing the NTP well-known port improves the ability of an attacker to perform blind/

off-path attacks (since knowledge of the port numbers is typically required for such attacks). A recent study [NIST-NTP] that analyzes the port numbers employed by NTP clients suggests that a considerable number of NTP clients employ the NTP well-known port as their local port, or select predictable ephemeral port numbers, thus improving the ability of attackers to perform blind/off-path attacks against NTP.

BCP 156 [RFC6056] already recommends the randomization of transport-protocol ephemeral ports. This document aligns NTP with the recommendation in BCP 156 [RFC6056], by formally updating [RFC5905] such that port randomization is employed for those NTP modes for which the use of the NTP well-known port is not needed.

2. Terminology

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. Considerations About Port Randomization in NTP

The following subsections analyze a number of considerations about transport-protocol ephemeral port randomization when applied to NTP.

3.1. Mitigation Against Off-path Attacks

There has been a fair share of work in the area of off-path/blind attacks against transport protocols and upper-layer protocols, such as [RFC5927] and [RFC4953]. Whether the target of the attack is a transport protocol instance (e.g., TCP connection) or an upper-layer protocol instance (e.g., an application protocol instance), the attacker is required to know or guess the five-tuple {Protocol, IP Source Address, IP Destination Address, Source Port, Destination Port} that identifies the target transport protocol instance or the transport protocol instance employed by the target upper-layer protocol instance. Therefore, increasing the difficulty of guessing this five-tuple helps mitigate blind/off-path attacks.

As a result of these considerations, BCP 156 [RFC6056] recommends the randomization of transport-protocol ephemeral ports. Thus, this document aims to bring the NTP specification [RFC5905] in line with the aforementioned recommendation.

We note that the use of port randomization is a transport-layer mitigation against off-path/blind attacks, and does not preclude (nor

is it precluded by) other possible mitigations for off-path attacks that might be implemented by an application protocol (e.g. [I-D.ietf-ntp-data-minimization]). For instance, some of the aforementioned mitigations may be ineffective against some off-path attacks [NTP-FRAG] or may benefit from the additional entropy provided by port randomization [NTP-security].

3.2. Effects on Path Selection

Intermediate systems implementing the Equal-Cost Multi-Path (ECMP) algorithm may select the outgoing link by computing a hash over a number of values, that include the transport-protocol source port. Thus, as discussed in [NTP-CHLNG], the selected client port may have an influence on the measured offset and delay.

If the source port is changed with each request, packets in different exchanges will be more likely to take different paths, which could cause the measurements to be less stable and have a negative impact on the stability of the clock.

Network paths to/from a given server are less likely to change between requests if port randomization is applied on a per-association basis. This approach minimizes the impact on the stability of NTP measurements, but may cause different clients in the same network synchronized to the same NTP server to have a significant stable offset between their clocks due to their NTP exchanges consistently taking different paths with different asymmetry in the network delay.

Section 4 recommends NTP implementations to randomize the ephemeral port number of client/server associations. The choice of whether to randomize the port number on a per-association or a per-request basis is left to the implementation.

3.3. Filtering of NTP traffic

In a number of scenarios (such as when mitigating DDoS attacks), a network operator may want to differentiate between NTP requests sent by clients, and NTP responses sent by NTP servers. If an implementation employs the NTP well-known port for the client port number, requests/responses cannot be readily differentiated by inspecting the source and destination port numbers. Implementation of port randomization for non-symmetrical modes allows for simple differentiation of NTP requests and responses, and for the enforcement of security policies that may be valuable for the mitigation of DDoS attacks, when all NTP clients in a given network employ port randomization.

3.4. Effect on NAT devices

Some NAT devices will not translate the source port of a packet when a privileged port number is employed. In networks where such NAT devices are employed, use of the NTP well-known port for the client port will essentially limit the number of hosts that may successfully employ NTP client implementations.

In the case of NAT devices that will translate the source port even when a privileged port is employed, packets reaching the external realm of the NAT will not employ the NTP well-known port as the local port, since the local port will normally be translated by the NAT device possibly, but not necessarily, with a random port.

3.5. Relation to Other Mitigations for Off-Path Attacks

Transport-protocol ephemeral port randomization is a best current practice (BCP 156) that helps mitigate off-path attacks at the transport-layer. It is orthogonal to other possible mitigations for off-path attacks that may be implemented at other layers (such as the use of timestamps in NTP) which may or may not be effective against some off-path attacks (see e.g. [NTP-FRAG]. This document aligns NTP with the existing best current practice on ephemeral port selection, irrespective of other techniques that may (and should) be implemented for mitigating off-path attacks.

4. Update to RFC5905

The following text from Section 9.1 ("Peer Process Variables") of [RFC5905]:

> dstport: UDP port number of the client, ordinarily the NTP port number PORT (123) assigned by the IANA. This becomes the source port number in packets sent from this association.

is replaced with:

> dstport: UDP port number of the client. In the case of broadcast server mode (5) and symmetric modes (1 and 2), it SHOULD contain the NTP port number PORT (123) assigned by the IANA. In the client mode (3), it SHOULD contain a randomized port number, as specified in [RFC6056]. The value in this variable becomes the source port number of packets sent from this association. The randomized port number SHOULD NOT be shared with other associations.
>
> If a client implementation performs ephemeral port randomization on a per-request basis, it SHOULD close the corresponding socket/

port after each request/response exchange. In order to prevent duplicate or delayed server packets from eliciting ICMP port unreachable error messages at the client, the client MAY wait for more responses from the server for a specific period of time (e.g. 3 seconds) before closing the UDP socket/port.

NOTES:

The choice of whether to randomize the ephemeral port number on a per-request or a per-association basis is left to the implementation, and should consider the possible effects on path selection along with its possible impact on time measurement.

On most current operating systems, which implement ephemeral port randomization [RFC6056], an NTP client may normally rely on the operating system to perform ephemeral port randomization. For example, NTP implementations using POSIX sockets may achieve ephemeral port randomization by *not* binding the socket with the bind() function, or binding it to port 0, which has a special meaning of "any port". connect()ing the socket will make the port inaccessible by other systems (that is, only packets from the specified remote socket will be received by the application).

5. Implementation Status

[RFC Editor: Please remove this section before publication of this document as an RFC.]

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 [RFC7942]. 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.

OpenNTPD:

[OpenNTPD] has never explicitly set the local port of NTP clients, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating

systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), OpenNTPD will employ port randomization for client ports.

chrony:
[chrony] by default does not set the local client port, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), chrony will employ port randomization for client ports.

nwtime.org's sntp client:
sntp does not explicitly set the local port, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), it will employ port randomization for client ports.

6. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

7. Security Considerations

The security implications of predictable numeric identifiers [I-D.irtf-pearg-numeric-ids-generation] (and of predictable transport-protocol port numbers [RFC6056] in particular) have been known for a long time now. However, the NTP specification has traditionally followed a pattern of employing common settings and code even when not strictly necessary, which at times has resulted in negative security and privacy implications (see e.g. [I-D.ietf-ntp-data-minimization]). The use of the NTP well-known port (123) for the srcport and dstport variables is not required for all operating modes. Such unnecessary usage comes at the expense of reducing the amount of work required for an attacker to successfully perform off-path/blind attacks against NTP. Therefore, this document formally updates [RFC5905], recommending the use of transport-protocol port randomization when use of the NTP well-known port is not required.

This issue has been tracked by US-CERT with VU#597821, and has been assigned CVE-2019-11331.

8. Acknowledgments

The authors would like to thank (in alphabetical order) Ivan Arce, Dhruv Dhody, Todd Glassey, Watson Ladd, Aanchal Malhotra, Danny Mayer, Gary E. Miller, Tomoyuki Sahara, Dieter Sibold, Steven Sommars, Kristof Teichel, and Ulrich Windl, for providing valuable comments on earlier versions of this document.

Watson Ladd raised the problem of DDoS mitigation when the NTP well-known port is employed as the client port (discussed in Section 3.3 of this document).

The authors would like to thank Harlan Stenn for answering questions about nwtime.org's NTP implementation.

Fernando would like to thank Nelida Garcia and Jorge Oscar Gont, for their love and support.

9. References

9.1. Normative References

[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>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <https://www.rfc-editor.org/info/rfc6056>.

[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

[chrony] "chrony", <https://chrony.tuxfamily.org/>.

[I-D.ietf-ntp-data-minimization]
Franke, D. and A. Malhotra, "NTP Client Data Minimization", draft-ietf-ntp-data-minimization-04 (work in progress), March 2019.

[I-D.irtf-pearg-numeric-ids-generation]
Gont, F. and I. Arce, "On the Generation of Transient Numeric Identifiers", draft-irtf-pearg-numeric-ids-generation-03 (work in progress), September 2020.

[NIST-NTP]
Sherman, J. and J. Levine, "Usage Analysis of the NIST Internet Time Service", Journal of Research of the National Institute of Standards and Technology Volume 121, March 2016, <https://tf.nist.gov/general/pdf/2818.pdf>.

[NTP-CHLNG]
Sommars, S., "Challenges in Time Transfer Using the Network Time Protocol (NTP)", Proceedings of the 48th Annual Precise Time and Time Interval Systems and Applications Meeting, Monterey, California pp. 271-290, January 2017, <http://leapsecond.com/ntp/NTP_Paper_Sommars_PTTI2017.pdf>.

[NTP-FRAG]
Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", NDSS'17, San Diego, CA. Feb 2017, 2017, <http://www.cs.bu.edu/~goldbe/papers/NTPattack.pdf>.

[NTP-security]
Malhotra, A., Van Gundy, M., Varia, V., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", Cryptology ePrint Archive Report 2016/1006, 2016, <https://eprint.iacr.org/2016/1006>.

[NTP-VULN]
Network Time Foundation, "Security Notice", Network Time Foundation's NTP Support Wiki , <https://support.ntp.org/bin/view/Main/SecurityNotice>.

[OpenNTPD]
"OpenNTPD Project", <https://www.openntpd.org>.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <https://www.rfc-editor.org/info/rfc4953>.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <https://www.rfc-editor.org/info/rfc5927>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

Authors' Addresses

Fernando Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: https://www.si6networks.com

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: ggont@si6networks.com
URI: https://www.si6networks.com

Miroslav Lichvar
Red Hat
Purkynova 115
Brno 612 00
Czech Republic

Email: mlichvar@redhat.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 25, 2021

A. Malhotra
Boston University
A. Langley
Google
W. Ladd
Cloudflare
M. Dansarie

February 21, 2021

Roughtime
draft-ietf-ntp-roughtime-04

Abstract

This document specifies Roughtime - a protocol that aims to achieve rough time synchronization while detecting servers that provide inaccurate time and providing cryptographic proof of their malfeasance.

Table of Contents

1. Introduction

Time synchronization is essential to Internet security as many security protocols and other applications require synchronization [RFC7384] [MCBG]. 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. Authenticating time servers prevents network adversaries from modifying time packets, but an authenticated time server still has full control over the contents of the time packet and may go rogue. The Roughtime protocol provides cryptographic proof of malfeasance, enabling clients to detect and prove to a third party a server's attempts to influence the time a client computes.

Protocol	Authenticated Server	Server Malfeasance Evidence
NTP, Chronos	N	N
NTP-MD5	Y*	N
NTP-Autokey	Y**	N
NTS	Y	N
Roughtime	Y	Y

Security Properties of current protocols

Table 1

Y* For security issues with symmetric-key based NTP-MD5 authentication, please refer to RFC 8573 [RFC8573].

Y** For security issues with Autokey Public Key Authentication, refer to [Autokey].

- If a server's timestamps do not fit into the time context of other servers' responses, then a Roughtime client can cryptographically prove this misbehavior to third parties. This helps detect "bad" servers.

- A Roughtime client can roughly detect (with no absolute guarantee) a delay attack [DelayAttacks] but can not cryptographically prove this to a third party. However, the absence of proof of malfeasance should not be considered a proof of absence of malfeasance. So Roughtime should not be used as a witness that a server is overall "good".

- Note that delay attacks cannot be detected/stopped by any protocol. Delay attacks can not, however, undermine the security guarantees provided by Roughtime.

- Although delay attacks cannot be prevented, they can be limited to a predetermined upper bound. This can be done by defining a maximal tolerable Round Trip Time (RTT) value, MAX-RTT, that a Roughtime client is willing to accept. A Roughtime client can measure the RTT of every request-response handshake and compare it

to MAX-RTT. If the RTT exceeds MAX-RTT, the corresponding server is assumed to be a falseticker. When this approach is used the maximal time error that can be caused by a delay attack is MAX-RTT/2. It should be noted that this approach assumes that the nature of the system is known to the client, including reasonable upper bounds on the RTT value.

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 [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 with a certificate rooted in a long-term public/private key pair over a value derived from a nonce provided by the client in its 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. For example, a radius of 1,000,000 microseconds means the server is absolutely confident that the true time is within one second of the reported time.

The server proves freshness of its response as follows: The client's request contains a nonce. The server incorporates the nonce into its signed response so that the client can verify the server's signatures covering the nonce issued by the client. 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 the transmission of the query and its reception. 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 of type 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.

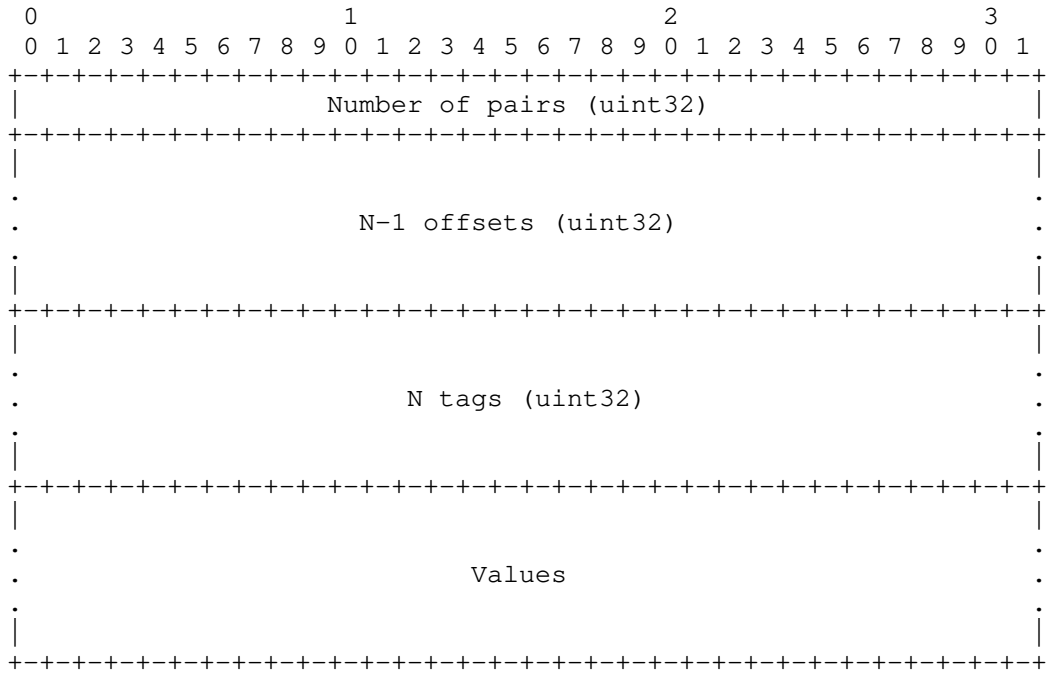


Figure 1: Roughtime Message Format

5.1. Data Types

5.1.1. int32

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

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 may also be listed as a sequence of up to four ASCII characters [RFC0020]. ASCII strings shorter than four characters can be unambiguously converted to tags by padding them with zero bytes. For example, the ASCII string "NONC" would correspond to the tag 0x434e4f4e and "PAD" would correspond to 0x00444150.

5.1.5. Timestamp

A timestamp is a uint64 interpreted in the following way. The most significant 3 bytes contain the integer part of a Modified Julian Date (MJD). The least significant 5 bytes is a count of the number of Coordinated Universal Time (UTC) microseconds [ITU-R_TF.460-6] since midnight on that day.

The MJD is the number of UTC days since 17 November 1858 [ITU-R_TF.457-2]. It is useful to note that 1 January 1970 is 40,587 days after 17 November 1858.

Note that, unlike NTP, this representation does not use the full number of bits in the fractional part and that days with leap seconds will have more or fewer than the nominal 86,400,000,000 microseconds.

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 ith tag begins at offset[i] and ends at offset[i+1]-1, with the exception of the last value which ends at the end of the message. Values may have zero length.

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

6. Protocol

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, while TCP transport is OPTIONAL.

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 5.1.

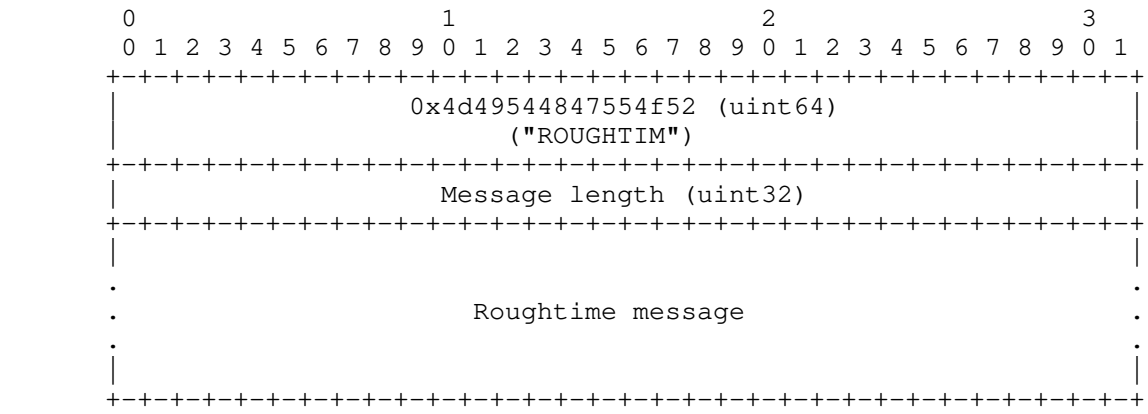


Figure 2: Roughtime Packet Format

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 [RFC0791] 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.

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 NONC and VER.

The value of the NONC tag is a 64 byte nonce. It SHOULD be generated in a manner indistinguishable from random.

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.

Tags other than NONC and VER SHOULD be ignored by the server.

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

6.2. Responses

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

The SIG tag is a signature over the SREP value using the public key contained in CERT, as explained below.

The SREP tag contains a time response. Its value is a Roughtime message with the tags ROOT, MIDP, and RADI. The server MAY include any of the tags DUT1, DTAI and LEAP in the contents of the SREP tag.

The NONC tag contains the nonce of the message being responded to.

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

The MIDP tag value is a timestamp of the moment of processing.

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

The DUT1 tag value is an int32 indicating the predicted difference between UT1 and UTC (UT1 - UTC) in milliseconds as given by the International Earth Rotation and Reference Systems Service (IERS).

The DTAI tag value is an int32 indicating the current difference between International Atomic Time (TAI) and UTC (TAI - UTC) in milliseconds as published in the International Bureau of Weights and Measures' (BIPM) Circular T.

The LEAP tag contains zero or more int32 values, each representing a past or future leap second event. Positive values represent the addition of a second and negative values represent the removal of a second. The absolute value represents the MJD of the second after the leap second event, i.e., the first second with a new UTC - TAI difference. The leap second events MUST be sorted in reverse

chronological order and the first item MUST be the last (past or future) leap second event that the server knows about. A LEAP tag with zero int32 values indicates that the server does not hold any updated leap second information.

The SIG tag value is a 64 byte Ed25519 signature [RFC8032] over a signature context concatenated with the entire value of a DELE or SREP tag. Signatures of DELE tags MUST use the ASCII string "RoughTime v1 delegation signature--" and signatures of SREP tags MUST use the ASCII string "RoughTime v1 response signature" as signature context. Both strings MUST include a terminating zero byte.

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 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.

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.

The PATH tag value is a multiple of 32 bytes long and represents a path of 32 byte hash values in the Merkle tree used to generate the ROOT value as described in 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 is zero.

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.

6.3. 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 of one request that a response message is sent in reply to. 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 [SHS]. 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 is stored in the PATH tag so that the client can reconstruct and validate the value in the ROOT tag using its nonce.

6.3.1. Root Value Validity Check Algorithm

One starts by computing the hash of the NONC value from the request, with 0x00 prepended. Then one walks from the least significant bit of INDX to the most significant bit, and also walks towards the end of PATH.

If PATH ends then the remaining bits of the INDX MUST be all zero. This indicates the termination of the walk, and the current value MUST equal ROOT if the response is valid.

If the current bit is 0, one hashes 0x01, the current hash, and the value from PATH to derive the next current value.

If the current bit is 1 one hashes 0x01, the value from PATH, and the current hash to derive the next current value.

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 guarantees 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 t1, we know the true time is in [t1-delta-sigma, t1-delta+sigma]. After d seconds have elapsed we know the true time is within [t1-delta-sigma-d*PHI, t1-delta+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.

8. Grease

Servers MAY 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 Servers

The below list contains a list of servers with their public keys in Base64 format. These servers may implement older versions of this specification.

```
address:       roughtime.cloudflare.com
port:          2002
long-term key: gD63hSj3ScS+wuOeGrubXlq35N1c5Lby/S+T7MNTjxo=

address:       roughtime.int08h.com
port:          2002
long-term key: AW5uAoTSTDfG5NfY1bTh08GUnOqlRb+HVhbJ3ODJvsE=

address:       roughtime.sandbox.google.com
port:          2002
long-term key: etPaaIxcBMY1oUeGpwvPMCJMwlRVNxv51KK/tktoJTQ=

address:       roughtime.se
port:          2002
long-term key: S3AzfZJ5CjSdkJ21ZJGbxqdYP/SoE8fXKY0+aicsehI=
```

10. Acknowledgements

Thomas Peterson corrected multiple nits. Peter Loethberg (Lothberg), Tal Mizrahi, Ragnar Sundblad, Kristof Teichel, and the other members of the NTP working group contributed comments and suggestions.

11. IANA Considerations

11.1. 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 [RFC6335]:

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

11.2. 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]]

11.3. 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 (OPTIONAL): 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
0x00444150	PAD	[[this memo]]
0x00474953	SIG	[[this memo]]
0x00524556	VER	[[this memo]]
0x31545544	DUT1	[[this memo]]
0x434e4f4e	NONC	[[this memo]]
0x454c4544	DELE	[[this memo]]
0x48544150	PATH	[[this memo]]
0x49415444	DTAI	[[this memo]]
0x49444152	RADI	[[this memo]]
0x4b425550	PUBK	[[this memo]]
0x5041454c	LEAP	[[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]]

12. 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 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, and 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.

13. 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.

14. References

14.1. Normative References

[ITU-R_TF.457-2]
ITU-R, "Use of the Modified Julian Date by the Standard-Frequency and Time-Signal Services", ITU-R Recommendation TF.457-2, October 1997.

[ITU-R_TF.460-6]
ITU-R, "Standard-Frequency and Time-Signal Emissions", ITU-R Recommendation TF.460-6, February 2002.

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[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/info/rfc8032>.

[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/info/rfc8259>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard", DOI 10.6028/NIST.FIPS.180-4, FIPS 180-4, August 2015.

14.2. Informative References

[Autokey] Rottger, S., "Analysis of the NTP Autokey Procedures", 2012, <https://zero-entropy.de/autokey_analysis.pdf>.

[DelayAttacks] Mizrahi, T., "A Game Theoretic Analysis of Delay Attacks Against Time Synchronization Protocols", DOI 10.1109/ISPCS.2012.6336612, 2012, <https://ieeexplore.ieee.org/document/6336612>.

[MCBG] Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", 2015, <https://eprint.iacr.org/2015/1020>.

[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>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[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>.

[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>.

[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>.

[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>.

Appendix A. Terms and Abbreviations

ASCII American Standard Code for Information Interchange

IANA Internet Assigned Numbers Authority

JSON JavaScript Object Notation [RFC8259]

MJD Modified Julian Date

NTP Network Time Protocol [RFC5905]

NTS Network Time Security [RFC8915]

TAI International Atomic Time (Temps Atomique International) [ITU-R_TF.460-6]

TCP Transmission Control Protocol [RFC0793]

UDP User Datagram Protocol [RFC0768]

UT Universal Time [ITU-R_TF.460-6]

UTC Coordinated Universal Time [ITU-R_TF.460-6]

Authors' Addresses

Aanchal Malhotra
Boston University
111 Cummington Mall
Boston 02215
USA

Email: aanchal4@bu.edu

Adam Langley
Google

Email:
agl@google.com

Watson Ladd
Cloudflare
101 Townsend St
San Francisco
USA

Email: watsonbladd@gmail.com

Marcus Dansarie
Sweden

Email: marcus@dansarie.se
URI: https://orcid.org/0000-0001-9246-0263

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 25, 2021

W. Ladd
Cloudflare
M. Dansarie

February 21, 2021

Roughtime Ecosystem
draft-ietf-ntp-roughtime-ecosystem-00

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 August 25, 2021.

Table of Contents

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 seralized and verified, as well as APIs to access and submit reports of malfeasnce 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. Impeachement

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 likey a failure of the generator of the errneous 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., and W. Ladd, "Roughtime", draft-ietf-ntp-roughtime-03 (work in progress), August 2020.

Authors' Addresses

Watson Ladd
Cloudflare
101 Townsend St
San Francisco
USA

Email: watsonbladd@gmail.com

Marcus Dansarie
Sweden

Email: marcus@dansarie.se
URI: https://orcid.org/0000-0001-9246-0263

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2021

N. Wu
D. Dhody, Ed.
Huawei
A. Sinha, Ed.
A. Kumar S N
RtBrick Inc.
Y. Zhao
Ericsson

March 8, 2021

A YANG Data Model for NTP
draft-ietf-ntp-yang-data-model-15

Abstract

This document defines a YANG data model for Network Time Protocol (NTP) implementations. The data model includes configuration data and state data.

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.

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 September 9, 2021.

Table of Contents

1. Introduction

This document defines a YANG [RFC7950] data model for Network Time Protocol [RFC5905] implementations.

The data model covers configuration of system parameters of NTP, such as access rules, authentication and VPN Routing and Forwarding (VRF) binding, and also associations of NTP in different modes and per-interface parameters. It also provides information about running state of NTP implementations.

1.1. Operational State

NTP Operational State is included in the same tree as NTP configuration, consistent with Network Management Datastore Architecture (NMDA) [RFC8342]. NTP current state and statistics are also maintained in the operational state. The operational state also includes the NTP association state.

1.2. Terminology

The terminology used in this document is aligned to [RFC5905].

1.3. Tree Diagrams

A simplified graphical representation of the data model is used in this document. This document uses the graphical representation of data models defined in [RFC8340].

1.4. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
if	ietf-interfaces	[RFC8343]
sys	ietf-system	[RFC7317]
acl	ietf-access-control-list	[RFC8519]
rt-types	ietf-routing-types	[RFC8294]
nacm	ietf-netconf-acm	[RFC8341]

Table 1: Prefixes and corresponding YANG modules

1.5. References in the Model

Following documents are referenced in the model defined in this document –

Title	Reference
Network Time Protocol Version 4: Protocol and Algorithms Specification	[RFC5905]
Common YANG Data Types	[RFC6991]
A YANG Data Model for System Management	[RFC7317]
YANG Data Model for Key Chains	[RFC8177]
Common YANG Data Types for the Routing Area	[RFC8294]
Network Configuration Access Control Model	[RFC8341]
A YANG Data Model for Interface Management	[RFC8343]
YANG Data Model for Network Access Control Lists (ACLs)	[RFC8519]
Message Authentication Code for the Network Time Protocol	[RFC8573]
The AES-CMAC Algorithm	[RFC4493]
The MD5 Message-Digest Algorithm	[RFC1321]
US Secure Hash Algorithm 1 (SHA1)	[RFC3174]

Table 2: References in the YANG modules

2. NTP data model

This document defines the YANG module "ietf-ntp", which has the following condensed structure:

```
module: ietf-ntp
  +--rw ntp!
```

```
      +--rw port?                    inet:port-number {ntp-port}?
      +--rw refclock-master!
      |  +--rw master-stratum?   ntp-stratum
      +--rw authentication {authentication}?
      |  +--rw auth-enabled?          boolean
      |  +--rw authentication-keys* [key-id]
      |     +--rw key-id       uint32
      |     |  ...
      +--rw access-rules {access-rules}?
      |  +--rw access-rule* [access-mode]
      |     +--rw access-mode    identityref
      |     +--rw acl?           -> /acl:acls/acl/name
      +--ro clock-state
      |  +--ro system-status
      |     +--ro clock-state                 identityref
      |     +--ro clock-stratum               ntp-stratum
      |     +--ro clock-refid                 refid
      |     |  ...
      +--rw unicast-configuration* [address type]
      |       {unicast-configuration}?
      |  +--rw address           inet:ip-address
      |  +--rw type              identityref
      |  |  ...
      +--ro associations* [address local-mode isconfigured]
      |  +--ro address           inet:ip-address
      |  +--ro local-mode        identityref
      |  +--ro isconfigured      boolean
      |  |  ...
      |  +--ro ntp-statistics
      |        ...
      |  +--rw interface* [name]
      |     +--rw name                if:interface-ref
      |     +--rw broadcast-server! {broadcast-server}?
      |     |  ...
      |     +--rw broadcast-client! {broadcast-client}?
      |     +--rw multicast-server* [address] {multicast-server}?
      |     |  +--rw address
      |     |  |       rt-types:ip-multicast-group-address
      |     |  |  ...
      |     +--rw multicast-client* [address] {multicast-client}?
      |     |  +--rw address    rt-types:ip-multicast-group-address
      |     +--rw manycast-server* [address] {manycast-server}?
      |     |  +--rw address    rt-types:ip-multicast-group-address
      |     +--rw manycast-client* [address] {manycast-client}?
      |        +--rw address
      |        |       rt-types:ip-multicast-group-address
      |        |  ...
      +--ro ntp-statistics
```

```
        +--...
```

The full data model tree for the YANG module "ietf-ntp" is in Appendix A.

This data model defines one top-level container which includes both the NTP configuration and the NTP running state including access rules, authentication, associations, unicast configurations, interfaces, system status and associations.

3. Relationship with NTPv4-MIB

If the device implements the NTPv4-MIB [RFC5907], data nodes from YANG module can be mapped to table entries in NTPv4-MIB.

The following tables list the YANG data nodes with corresponding objects in the NTPv4-MIB.

YANG NTP Configuration Data Nodes and Related NTPv4-MIB Objects

YANG data nodes in /ntp/clock-state/system-status	NTPv4-MIB objects
clock-state	ntpEntStatusCurrentMode
clock-stratum	ntpEntStatusStratum
clock-refid	ntpEntStatusActiveRefSourceId
	ntpEntStatusActiveRefSourceName
clock-precision	ntpEntTimePrecision
clock-offset	ntpEntStatusActiveOffset
root-dispersion	ntpEntStatusDispersion

YANG data nodes in /ntp/associations/	NTPv4-MIB objects
address	ntpAssocAddressType
	ntpAssocAddress
stratum	ntpAssocStratum
refid	ntpAssocRefId
offset	ntpAssocOffset
delay	ntpAssocStatusDelay
dispersion	ntpAssocStatusDispersion
ntp-statistics/packet-sent	ntpAssocStatOutPkts
ntp-statistics/packet-received	ntpAssocStatInPkts
ntp-statistics/packet-dropped	ntpAssocStatProtocolError

YANG NTP State Data Nodes and Related NTPv4-MIB Objects

4. Relationship with RFC 7317

This section describes the relationship with NTP definition in Section 3.2 System Time Management of [RFC7317] . YANG data nodes in /ntp/ also support per-interface configuration which is not supported in /system/ntp. If the yang model defined in this document is implemented, then /system/ntp SHOULD NOT be used and MUST be ignored.

YANG data nodes in /ntp/	YANG data nodes in /system/ntp
ntp!	enabled
unicast-configuration	server
	server/name
unicast-configuration/address	server/transport/udp/address
unicast-configuration/port	server/transport/udp/port
unicast-configuration/type	server/association-type
unicast-configuration/iburst	server/iburst
unicast-configuration/prefer	server/prefer

YANG NTP Configuration Data Nodes and counterparts in RFC 7317 Objects

5. Access Rules

An Access Control List (ACL) is one of the basic elements used to configure device-forwarding behavior. An ACL is a user-ordered set of rules that is used to filter traffic on a networking device.

As per [RFC1305] and [RFC5905], NTP could include an access-control feature that prevents unauthorized access and controls which peers are allowed to update the local clock. Further it is useful to differentiate between the various kinds of access and attach a different acl-rule to each. For this, the YANG module allows such configuration via /ntp/access-rules. The access-rule itself is configured via [RFC8519].

Following access modes are supported -

- Peer: Permit others to synchronize their time with the NTP entity or it can synchronize its time with others. NTP control queries are also accepted.

- Server: Permit others to synchronize their time with the NTP entity, but vice versa is not supported. NTP control queries are accepted.

- Server-only: Permit others to synchronize their time with NTP entity, but vice versa is not supported. NTP control queries are not accepted.

- Query-only: Only control queries are accepted.

Query-only is the most restricted where as the peer is the full access authority. The ability to give different ACL rules for different access modes allows for a greater control by the operator.

6. Key Management

As per [RFC1305] and [RFC5905], when authentication is enabled, NTP employs a crypto-checksum, computed by the sender and checked by the receiver, together with a set of predistributed algorithms, and cryptographic keys indexed by a key identifier included in the NTP message. This key-id is a 32-bit unsigned integer that MUST be configured on the NTP peers before the authentication could be used. For this reason, this YANG module allows such configuration via /ntp/authentication/authentication-keys/. Further at the time of configuration of NTP association (for example unicast-server), the key-id is specified.

7. NTP Version

This YANG model allow a version to be configured for the NTP association i.e. an operator can control the use of NTPv3 [RFC1305] or NTPv4 [RFC5905] for each association it forms. This allows backward compatibility with a legacy system.

8. NTP YANG Module

```
<CODE BEGINS> file "ietf-ntp@2021-03-09.yang"
module ietf-ntp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ntp";
  prefix ntp;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-inet-types {
```

```
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ietf-system {
    prefix sys;
    reference
      "RFC 7317: A YANG Data Model for System Management";
  }
  import ietf-access-control-list {
    prefix acl;
    reference
      "RFC 8519: YANG Data Model for Network Access Control
       Lists (ACLs)";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Protocol (NETCONF) Access
       Control Model";
  }

  organization
    "IETF NTP (Network Time Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/ntp/>
     WG List:  <mailto: ntp@ietf.org
     Editor:   Dhruv Dhody
               <mailto:dhruv.ietf@gmail.com>
     Editor:   Ankit Kumar Sinha
               <mailto:ankit.ietf@gmail.com>";
  description
    "This document defines a YANG data model for Network Time Protocol
     (NTP) implementations. The data model includes configuration data
     and state data.

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

```
     Redistribution and use in source and binary forms,
     with or without modification, is permitted pursuant to,
     and subject to the license terms contained in, the
     Simplified BSD License set forth in Section 4.c of the
     IETF Trust's Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.

     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) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";

  revision 2021-03-09 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for NTP.";
  }

  /* Note: The RFC Editor will replace XXXX with the number assigned
  to this document once it becomes an RFC.*/
  /* Typedef Definitions */

  typedef ntp-stratum {
    type uint8 {
      range "1..16";
    }
    description
      "The level of each server in the hierarchy is defined by
       a stratum. Primary servers are assigned with stratum
       one; secondary servers at each lower level are assigned with
       one stratum greater than the preceding level";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3";
  }

  typedef ntp-version {
    type uint8;
    default "4";
    description
      "The current NTP version supported by corresponding
       association.";
    reference
```

```
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 1";
  }

  typedef refid {
    type union {
      type inet:ipv4-address;
      type uint32;
      type string {
        length "4";
      }
    }
    description
      "A code identifying the particular server or reference
       clock.  The interpretation depends upon stratum. It
       could be an IPv4 address or first 32 bits of the MD5 hash of
       the IPv6 address or a string for the Reference Identifier
       and KISS codes. Some examples:
       -- a refclock ID like '127.127.1.0' for local clock sync
       -- uni/multi/broadcast associations for IPv4 will look like
       '203.0.113.1' and '0x4321FEDC' for IPv6
       -- sync with primary source will look like 'DCN', 'NIST',
       'ATOM'
       -- KISS codes will look like 'AUTH', 'DROP', 'RATE'
       Note that the use of MD5 hash for IPv6 address is not for
       cryptographic purposes ";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 7.3";
  }

  typedef ntp-date-and-time {
    type union {
      type yang:date-and-time;
      type uint8;
    }
    description
      "Follows the normal date-and-time format when valid value
       exist, otherwise allows for setting special value such as
       zero.";
  }

  /* features */

  feature ntp-port {
    description
      "Support for NTP port configuration";
    reference
```

```
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 7.2";
  }

  feature authentication {
    description
      "Support for NTP symmetric key authentication";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 7.3";
  }

  feature deprecated {
    description
      "Support deprecated MD5-based authentication (RFC 8573) or
       SHA-1 or any other deprecated authentication.
       It is enabled to support legacy compatibility when secure
       cryptographic algorithm is not availaible to use.";
    reference
      "RFC 1321: The MD5 Message-Digest Algorithm
       RFC 3174: US Secure Hash Algorithm 1 (SHA1)";
  }

  feature hex-key-string {
    description
      "Support hexadecimal key string.";
  }

  feature access-rules {
    description
      "Support for NTP access control";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 9.2";
  }

  feature unicast-configuration {
    description
      "Support for NTP client/server or active/passive
       in unicast";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3";
  }

  feature broadcast-server {
    description
      "Support for broadcast server";
```

```
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3";
  }

  feature broadcast-client {
    description
      "Support for broadcast client";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3";
  }

  feature multicast-server {
    description
      "Support for multicast server";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3.1";
  }

  feature multicast-client {
    description
      "Support for multicast client";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3.1";
  }

  feature manycast-server {
    description
      "Support for manycast server";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3.1";
  }

  feature manycast-client {
    description
      "Support for manycast client";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3.1";
  }

  /* Identity */
  /* unicast-configurations types */
```

```
  identity unicast-configuration-type {
    if-feature "unicast-configuration";
    description
      "This defines NTP unicast mode of operation as used
       for unicast-configurations.";
  }

  identity uc-server {
    if-feature "unicast-configuration";
    base unicast-configuration-type;
    description
      "Use client association mode. This device
       will not provide synchronization to the
       configured NTP server.";
  }

  identity uc-peer {
    if-feature "unicast-configuration";
    base unicast-configuration-type;
    description
      "Use symmetric active association mode.
       This device may provide synchronization
       to the configured NTP server.";
  }

  /* association-modes */

  identity association-mode {
    description
      "The NTP association modes.";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 3";
  }

  identity active {
    base association-mode;
    description
      "Use symmetric active association mode (mode 1).
       This device may synchronize with its NTP peer,
       or provide synchronization to configured NTP peer.";
  }

  identity passive {
    base association-mode;
    description
      "Use symmetric passive association mode (mode 2).
       This device has learned this association dynamically.
```

```
       This device may synchronize with its NTP peer.";
  }

  identity client {
    base association-mode;
    description
      "Use client association mode (mode 3).
       This device will not provide synchronization
       to the configured NTP server.";
  }

  identity server {
    base association-mode;
    description
      "Use server association mode (mode 4).
       This device will provide synchronization to
       NTP clients.";
  }

  identity broadcast-server {
    base association-mode;
    description
      "Use broadcast server mode (mode 5).
       This mode defines that its either working
       as broadcast-server or multicast-server.";
  }

  identity broadcast-client {
    base association-mode;
    description
      "This mode defines that its either working
       as broadcast-client (mode 6) or multicast-client.";
  }

  /* access-mode */

  identity access-mode {
    if-feature "access-rules";
    description
      "This defines NTP access modes. These identifies
       how the ACL is applied with NTP.";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 9.2";
  }

  identity peer-access-mode {
    if-feature "access-rules";
```

```
    base access-mode;
    description
      "Permit others to synchronize their time with this NTP
       entity or it can synchronize its time with others.
       NTP control queries are also accepted. This enables
       full access authority.";
  }

  identity server-access-mode {
    if-feature "access-rules";
    base access-mode;
    description
      "Permit others to synchronize their time with this NTP
       entity, but vice versa is not supported. NTP control
       queries are accepted.";
  }

  identity server-only-access-mode {
    if-feature "access-rules";
    base access-mode;
    description
      "Permit others to synchronize their time with this NTP
       entity, but vice versa is not supported. NTP control
       queries are not accepted.";
  }

  identity query-only-access-mode {
    if-feature "access-rules";
    base access-mode;
    description
      "Only control queries are accepted.";
  }

  /* clock-state */

  identity clock-state {
    description
      "This defines NTP clock status at a high level.";
  }

  identity synchronized {
    base clock-state;
    description
      "Indicates that the local clock has been synchronized with
       an NTP server or the reference clock.";
  }

  identity unsynchronized {
```

```
    base clock-state;
    description
      "Indicates that the local clock has not been synchronized
       with any NTP server.";
  }

  /* ntp-sync-state */

  identity ntp-sync-state {
    description
      "This defines NTP clock sync state at a more granular
       level. Referred as 'Clock state definitions' in RFC 5905";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Appendix A.1.1";
  }

  identity clock-not-set {
    base ntp-sync-state;
    description
      "Indicates the clock is not updated.";
  }

  identity freq-set-by-cfg {
    base ntp-sync-state;
    description
      "Indicates the clock frequency is set by
       NTP configuration or file.";
  }

  identity spike {
    base ntp-sync-state;
    description
      "Indicates a spike is detected.";
  }

  identity freq {
    base ntp-sync-state;
    description
      "Indicates the frequency mode.";
  }

  identity clock-synchronized {
    base ntp-sync-state;
    description
      "Indicates that the clock is synchronized";
  }
```

```
  /* crypto-algorithm */

  identity crypto-algorithm {
    description
      "Base identity of cryptographic algorithm options.";
  }

  identity hmac-sha-1-12 {
    base crypto-algorithm;
    description
      "The HMAC-SHA1-12 algorithm.";
  }

  identity md5 {
    if-feature "deprecated";
    base crypto-algorithm;
    description
      "The MD5 algorithm. Note that RFC 8573
       deprecates the use of MD5-based authentication.";
  }

  identity sha-1 {
    if-feature "deprecated";
    base crypto-algorithm;
    description
      "The SHA-1 algorithm.";
  }

  identity hmac-sha-1 {
    base crypto-algorithm;
    description
      "HMAC-SHA-1 authentication algorithm.";
  }

  identity hmac-sha-256 {
    description
      "HMAC-SHA-256 authentication algorithm.";
  }

  identity hmac-sha-384 {
    description
      "HMAC-SHA-384 authentication algorithm.";
  }

  identity hmac-sha-512 {
    description
      "HMAC-SHA-512 authentication algorithm.";
  }
```

```
  identity aes-cmac {
    base crypto-algorithm;
    description
      "The AES-CMAC algorithm - required by
       RFC 8573 for MAC for the NTP";
    reference
      "RFC 4493: The AES-CMAC Algorithm";
  }

  /* Groupings */

  grouping key {
    description
      "The key.";
    nacm:default-deny-all;
    choice key-string-style {
      description
        "Key string styles";
      case keystring {
        leaf keystring {
          type string;
          description
            "Key string in ASCII format.";
        }
      }
      case hexadecimal {
        if-feature "hex-key-string";
        leaf hexadecimal-string {
          type yang:hex-string;
          description
            "Key in hexadecimal string format.  When compared
             to ASCII, specification in hexadecimal affords
             greater key entropy with the same number of
             internal key-string octets.  Additionally, it
             discourages usage of well-known words or
             numbers.";
        }
      }
    }
  }

  grouping authentication-key {
    description
      "To define an authentication key for a Network Time
       Protocol (NTP) time source.";
    nacm:default-deny-all;
    leaf key-id {
      type uint32 {
```

```
        range "1..max";
      }
      description
        "Authentication key identifier.";
    }
    leaf algorithm {
      type identityref {
        base crypto-algorithm;
      }
      description
        "Authentication algorithm. Note that RFC 8573
         deprecates the use of MD5-based authentication
         and recommends AES-CMAC.";
    }
    container key {
      uses key;
      description
        "The key. Note that RFC 8573 deprecates the use
         of MD5-based authentication.";
    }
    leaf istrusted {
      type boolean;
      description
        "Key-id is trusted or not";
    }
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification, Section 7.3";
  }

  grouping authentication {
    description
      "Authentication.";
    choice authentication-type {
      description
        "Type of authentication.";
      case symmetric-key {
        leaf key-id {
          type leafref {
            path "/ntp:ntp/ntp:authentication/"
               + "ntp:authentication-keys/ntp:key-id";
          }
          description
            "Authentication key id referenced in this
             association.";
        }
      }
    }
```

```
  }

  grouping statistics {
    description
      "NTP packet statistic.";
    leaf discontinuity-time {
      type ntp-date-and-time;
      description
        "The time on the most recent occasion at which any one or
         more of this NTP counters suffered a discontinuity. If
         no such discontinuities have occurred, then this node
         contains the time the NTP association was
         (re-)initialized.";
    }
    leaf packet-sent {
      type yang:counter32;
      description
        "The total number of NTP packets delivered to the
         transport service by this NTP entity for this
         association.
         Discontinuities in the value of this counter can occur
         upon cold start or reinitialization of the NTP entity, the
         management system and at other times.";
    }
    leaf packet-sent-fail {
      type yang:counter32;
      description
        "The number of times NTP packets sending failed.";
    }
    leaf packet-received {
      type yang:counter32;
      description
        "The total number of NTP packets delivered to the
         NTP entity from this association.
         Discontinuities in the value of this counter can occur
         upon cold start or reinitialization of the NTP entity, the
         management system and at other times.";
    }
    leaf packet-dropped {
      type yang:counter32;
      description
        "The total number of NTP packets that were delivered
         to this NTP entity from this association and this entity
         was not able to process due to an NTP protocol error.
         Discontinuities in the value of this counter can occur
         upon cold start or reinitialization of the NTP entity, the
         management system and at other times.";
    }
```

```
  }

  grouping common-attributes {
    description
      "NTP common attributes for configuration.";
    leaf minpoll {
      type int8;
      default "6";
      description
        "The minimum poll interval used in this association.";
      reference
        "RFC 5905: Network Time Protocol Version 4: Protocol and
         Algorithms Specification, Section 7.2";
    }
    leaf maxpoll {
      type int8;
      default "10";
      description
        "The maximum poll interval used in this association.";
      reference
        "RFC 5905: Network Time Protocol Version 4: Protocol and
         Algorithms Specification, Section 7.2";
    }
    leaf port {
      if-feature "ntp-port";
      type inet:port-number {
        range "123 | 1025..max";
      }
      default "123";
      description
        "Specify the port used to send NTP packets.";
      reference
        "RFC 5905: Network Time Protocol Version 4: Protocol and
         Algorithms Specification, Section 7.2";
    }
    leaf version {
      type ntp-version;
      description
        "NTP version.";
    }
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
       Algorithms Specification";
  }

  grouping association-ref {
    description
      "Reference to NTP association mode";
```

```
    leaf associations-address {
      type leafref {
        path "/ntp:ntp/ntp:associations/ntp:address";
      }
      description
        "Indicates the association's address
         which result in clock synchronization.";
    }
    leaf associations-local-mode {
      type leafref {
        path "/ntp:ntp/ntp:associations/ntp:local-mode";
      }
      description
        "Indicates the association's local-mode
         which result in clock synchronization.";
    }
    leaf associations-isconfigured {
      type leafref {
        path "/ntp:ntp/ntp:associations/"
           + "ntp:isconfigured";
      }
      description
        "The association was configured or dynamic
         which result in clock synchronization.";
    }
  }

  /* Configuration data nodes */

  container ntp {
    when 'false() = boolean(/sys:system/sys:ntp)' {
      description
        "Applicable when the system /sys/ntp/ is not used.";
    }
    presence "NTP is enabled and system should attempt to
              synchronize the system clock with an NTP server
              from the 'ntp/associations' list.";
    description
      "Configuration parameters for NTP.";
    leaf port {
      if-feature "ntp-port";
      type inet:port-number {
        range "123 | 1025..max";
      }
      default "123";
      description
        "Specify the port used to send and receive NTP packets.";
      reference
```

```
        "RFC 5905: Network Time Protocol Version 4: Protocol and
         Algorithms Specification, Section 7.2";
    }
    container refclock-master {
      presence "NTP master clock is enabled.";
      description
        "Configures the local clock of this device as NTP server.";
      leaf master-stratum {
        type ntp-stratum;
        default "16";
        description
          "Stratum level from which NTP clients get their time
           synchronized.";
      }
    }
    container authentication {
      if-feature "authentication";
      description
        "Configuration of authentication.";
      leaf auth-enabled {
        type boolean;
        default "false";
        description
          "Controls whether NTP authentication is enabled
           or disabled on this device.";
      }
      list authentication-keys {
        key "key-id";
        uses authentication-key;
        description
          "List of authentication keys.";
      }
    }
    container access-rules {
      if-feature "access-rules";
      description
        "Configuration to control access to NTP service
         by using NTP access-group feature.
         The access-mode identifies how the acl is
         applied with NTP.";
      list access-rule {
        key "access-mode";
        description
          "List of access rules.";
        leaf access-mode {
          type identityref {
            base access-mode;
          }
```

```
          description
            "NTP access mode. The definition of each possible value:
             peer: Both time request and control query can be
             performed.
             server: Enables the server access and query.
             synchronization: Enables the server access only.
             query: Enables control query only.";
        }
        leaf acl {
          type leafref {
            path "/acl:acls/acl:acl/acl:name";
          }
          description
            "Control access configuration to be used.";
        }
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 9.2";
      }
    }
    container clock-state {
      config false;
      description
        "Clock operational state of the NTP.";
      container system-status {
        description
          "System status of NTP.";
        leaf clock-state {
          type identityref {
            base clock-state;
          }
          mandatory true;
          description
            "The state of system clock. The definition of each
             possible value is:
             synchronized: Indicates local clock is synchronized.
             unsynchronized: Indicates local clock is not
             synchronized.";
        }
        leaf clock-stratum {
          type ntp-stratum;
          mandatory true;
          description
            "The NTP entity's own stratum value. Should be one greater
             than preceeding level. 16 if unsyncronized.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3";
```

```
        }
        leaf clock-refid {
          type refid;
          mandatory true;
          description
            "A code identifying the particular server or reference
             clock.  The interpretation depends upon stratum. It
             could be an IPv4 address or first 32 bits of the MD5 hash
             of the IPv6 address or a string for the Reference
             Identifier and KISS codes. Some examples:
             -- a refclock ID like '127.127.1.0' for local clock sync
             -- uni/multi/broadcast associations for IPv4 will look like
             '203.0.113.1' and '0x4321FEDC' for IPv6
             -- sync with primary source will look like 'DCN', 'NIST',
             'ATOM'
             -- KISS codes will look like 'AUTH', 'DROP', 'RATE'
             Note that the use of MD5 hash for IPv6 address is not for
             cryptographic purposes ";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 7.3";
        }
        uses association-ref {
          description
            "Reference to Association.";
        }
        leaf nominal-freq {
          type decimal64 {
            fraction-digits 4;
          }
          units "Hz";
          mandatory true;
          description
            "The nominal frequency of the local clock. An ideal
             frequency with zero uncertainty.";
        }
        leaf actual-freq {
          type decimal64 {
            fraction-digits 4;
          }
          units "Hz";
          mandatory true;
          description
            "The actual frequency of the local clock.";
        }
        leaf clock-precision {
          type int8;
          units "Hz";
```

```
          mandatory true;
          description
            "Clock precision of this system in integer format
             (prec=2^(-n)). A value of 5 would mean 2^-5 = 31.25 ms.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 7.3";
        }
        leaf clock-offset {
          type decimal64 {
            fraction-digits 3;
          }
          units "milliseconds";
          description
            "The time offset to the current selected reference time
             source e.g., '0.032ms' or '1.232ms'.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 9.1";
        }
        leaf root-delay {
          type decimal64 {
            fraction-digits 3;
          }
          units "milliseconds";
          description
            "Total delay along the path to root clock.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 4 and 7.3";
        }
        leaf root-dispersion {
          type decimal64 {
            fraction-digits 3;
          }
          units "milliseconds";
          description
            "The dispersion between the local clock
             and the root clock, e.g., '6.927ms'.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 4 and 7.3";
        }
        leaf reference-time {
          type ntp-date-and-time;
          description
            "The reference timestamp. Time when the system clock was
             last set or corrected";
```

```
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 7.3";
        }
        leaf sync-state {
          type identityref {
            base ntp-sync-state;
          }
          mandatory true;
          description
            "The synchronization status of the local clock. Referred to
             as 'Clock state definitions' in RFC 5905";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Appendix A.1.1";
        }
      }
    }
    list unicast-configuration {
      if-feature "unicast-configuration";
      key "address type";
      description
        "List of NTP unicast-configurations.";
      leaf address {
        type inet:ip-address;
        description
          "Address of this association.";
      }
      leaf type {
        type identityref {
          base unicast-configuration-type;
        }
        description
          "Use client association mode.  This device
           will not provide synchronization to the
           configured NTP server.";
      }
      container authentication {
        if-feature "authentication";
        description
          "Authentication used for this association.";
        uses authentication;
      }
      leaf prefer {
        type boolean;
        default "false";
        description
          "Whether this association is preferred or not.";
```

```
      }
      leaf burst {
        type boolean;
        default "false";
        description
          "If set, a series of packets are sent instead of a single
           packet within each synchronization interval to achieve
           faster synchronization.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 13.1";
      }
      leaf iburst {
        type boolean;
        default "false";
        description
          "If set, a series of packets are sent instead of a single
           packet within the initial synchronization interval to
           achieve faster initial synchronization.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 13.1";
      }
      leaf source {
        type if:interface-ref;
        description
          "The interface whose IP address is used by this association
           as the source address.";
      }
      uses common-attributes {
        description
          "Common attributes like port, version, min and max
           poll.";
      }
    }
    list associations {
      key "address local-mode isconfigured";
      config false;
      description
        "List of NTP associations. Here address, local-mode
         and isconfigured are required to uniquely identify
         a particular association. Lets take following examples -

         1) If RT1 acting as broadcast server,
         and RT2 acting as broadcast client, then RT2
         will form dynamic association with address as RT1,
         local-mode as client and isconfigured as false.
```

```
          2) When RT2 is configured
          with unicast-server RT1, then RT2 will form
          association with address as RT1, local-mode as client
          and isconfigured as true.

          Thus all 3 leaves are needed as key to unique identify
          the association.";
        leaf address {
          type inet:ip-address;
          description
            "The address of this association. Represents the IP
             address of a unicast/multicast/broadcast address.";
        }
        leaf local-mode {
          type identityref {
            base association-mode;
          }
          description
            "Local mode of this NTP association.";
        }
        leaf isconfigured {
          type boolean;
          description
            "Indicates if this association is configured or
             dynamically learned.";
        }
        leaf stratum {
          type ntp-stratum;
          description
            "The association stratum value.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3";
        }
        leaf refid {
          type refid;
          description
            "A code identifying the particular server or reference
             clock.  The interpretation depends upon stratum. It
             could be an IPv4 address or first 32 bits of the MD5 hash of
             the IPv6 address or a string for the Reference Identifier
             and KISS codes. Some examples:
             -- a refclock ID like '127.127.1.0' for local clock sync
             -- uni/multi/broadcast associations for IPv4 will look like
             '203.0.113.1' and '0x4321FEDC' for IPv6
             -- sync with primary source will look like 'DCN', 'NIST',
             'ATOM'
             -- KISS codes will look like 'AUTH', 'DROP', 'RATE'
```

```
          Note that the use of MD5 hash for IPv6 address is not for
          cryptographic purposes";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
          Algorithms Specification, Section 7.3";
      }
      leaf authentication {
        if-feature "authentication";
        type leafref {
          path "/ntp:ntp/ntp:authentication/"
             + "ntp:authentication-keys/ntp:key-id";
        }
        description
          "Authentication Key used for this association.";
      }
      leaf prefer {
        type boolean;
        default "false";
        description
          "Indicates if this association is preferred.";
      }
      leaf peer-interface {
        type if:interface-ref;
        description
          "The interface which is used for communication.";
      }
      uses common-attributes {
        description
          "Common attributes like port, version, min and
          max poll.";
      }
      leaf reach {
        type uint8;
        description
          "The reachability of the configured
          server or peer.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
          Algorithms Specification, Section 9.2 and 13";
      }
      leaf unreach {
        type uint8;
        description
          "The unreachability of the configured
          server or peer.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
          Algorithms Specification, Section 9.2 and 13";
```

```
      }
      leaf poll {
        type int8;
        units "seconds";
        description
          "The polling interval for current association";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 7.3";
      }
      leaf now {
        type uint32;
        units "seconds";
        description
          "The time since the last NTP packet was
           received or last synchronized.";
      }
      leaf offset {
        type decimal64 {
          fraction-digits 3;
        }
        units "milliseconds";
        description
          "The offset between the local clock
           and the peer clock, e.g., '0.032ms' or '1.232ms'";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 8";
      }
      leaf delay {
        type decimal64 {
          fraction-digits 3;
        }
        units "milliseconds";
        description
          "The network delay between the local clock
           and the peer clock.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 8";
      }
      leaf dispersion {
        type decimal64 {
          fraction-digits 3;
        }
        units "milliseconds";
        description
          "The root dispersion between the local clock
```

```
           and the peer clock.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 10";
      }
      leaf originate-time {
        type ntp-date-and-time;
        description
          "This is the local time, in timestamp format,
           when latest NTP packet was sent to peer (called T1).";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 8";
      }
      leaf receive-time {
        type ntp-date-and-time;
        description
          "This is the local time, in timestamp format,
           when latest NTP packet arrived at peer (called T2).
           If the peer becomes unreachable the value is set to zero.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 8";
      }
      leaf transmit-time {
        type ntp-date-and-time;
        description
          "This is the local time, in timestamp format,
           at which the NTP packet departed the peer (called T3).
           If the peer becomes unreachable the value is set to zero.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 8";
      }
      leaf input-time {
        type ntp-date-and-time;
        description
          "This is the local time, in timestamp format,
           when the latest NTP message from the peer arrived (called
           T4). If the peer becomes unreachable the value is set to
           zero.";
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 8";
      }
      container ntp-statistics {
        description
          "Per Peer packet send and receive statistics.";
```

```
        uses statistics {
          description
            "NTP send and receive packet statistics.";
        }
      }
    }
    container interfaces {
      description
        "Configuration parameters for NTP interfaces.";
      list interface {
        key "name";
        description
          "List of interfaces.";
        leaf name {
          type if:interface-ref;
          description
            "The interface name.";
        }
        container broadcast-server {
          if-feature "broadcast-server";
          presence "NTP broadcast-server is configured";
          description
            "Configuration of broadcast server.";
          leaf ttl {
            type uint8;
            description
              "Specifies the time to live (TTL) for a
               broadcast packet.";
            reference
              "RFC 5905: Network Time Protocol Version 4: Protocol and
               Algorithms Specification, Section 3.1";
          }
          container authentication {
            if-feature "authentication";
            description
              "Authentication used for this association.";
            uses authentication;
          }
          uses common-attributes {
            description
              "Common attributes such as port, version, min and
               max poll.";
          }
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3.1";
        }
        container broadcast-client {
```

```
          if-feature "broadcast-client";
          presence "NTP broadcast-client is configured.";
          description
            "Configuration of broadcast-client.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3.1";
        }
        list multicast-server {
          if-feature "multicast-server";
          key "address";
          description
            "Configuration of multicast server.";
          leaf address {
            type rt-types:ip-multicast-group-address;
            description
              "The IP address to send NTP multicast packets.";
          }
          leaf ttl {
            type uint8;
            description
              "Specifies the time to live (TTL) for a
               multicast packet.";
            reference
              "RFC 5905: Network Time Protocol Version 4: Protocol and
               Algorithms Specification, Section 3.1";
          }
          container authentication {
            if-feature "authentication";
            description
              "Authentication used for this association.";
            uses authentication;
          }
          uses common-attributes {
            description
              "Common attributes such as port, version, min and
               max poll.";
          }
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3.1";
        }
        list multicast-client {
          if-feature "multicast-client";
          key "address";
          description
            "Configuration of multicast-client.";
          leaf address {
```

```
            type rt-types:ip-multicast-group-address;
            description
              "The IP address of the multicast group to
               join.";
          }
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3.1";
        }
        list manycast-server {
          if-feature "manycast-server";
          key "address";
          description
            "Configuration of manycast server.";
          leaf address {
            type rt-types:ip-multicast-group-address;
            description
              "The multicast group IP address to receive
               manycast client messages.";
          }
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 3.1";
        }
        list manycast-client {
          if-feature "manycast-client";
          key "address";
          description
            "Configuration of manycast-client.";
          leaf address {
            type rt-types:ip-multicast-group-address;
            description
              "The group IP address that the manycast client
               broadcasts the request message to.";
          }
          container authentication {
            if-feature "authentication";
            description
              "Authentication used for this association.";
            uses authentication;
          }
          leaf ttl {
            type uint8;
            description
              "Specifies the maximum time to live (TTL) for
               the expanding ring search.";
            reference
              "RFC 5905: Network Time Protocol Version 4: Protocol and
```

```
            Algorithms Specification, Section 3.1";
        }
        leaf minclock {
          type uint8;
          description
            "The minimum manycast survivors in this
             association.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 13.2";
        }
        leaf maxclock {
          type uint8;
          description
            "The maximum manycast candidates in this
             association.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 13.2";
        }
        leaf beacon {
          type int8;
          units "seconds";
          description
            "The beacon is the upper limit of poll interval. When the
             ttl reaches its limit without finding the minimum number
             of manycast servers, the poll interval increases until
             reaching the beacon value, when it starts over from the
             beginning.";
          reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
             Algorithms Specification, Section 13.2";
        }
        uses common-attributes {
          description
            "Common attributes like port, version, min and
             max poll.";
        }
        reference
          "RFC 5905: Network Time Protocol Version 4: Protocol and
           Algorithms Specification, Section 3.1";
      }
    }
  }
  container ntp-statistics {
    config false;
    description
      "Total NTP packet statistics.";
```

```
      uses statistics {
        description
          "NTP send and receive packet statistics.";
      }
    }
  }
}
<CODE ENDS>
```

9. Usage Example

This section include examples for illustration purposes.

Note: '\' line wrapping per [RFC8792].

9.1. Unicast association

This example describes how to configure a preferred unicast server present at 192.0.2.1 running at port 1025 with authentication-key 10 and version 4 (default).

```
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <running/>
    </target>
    <config>
      <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
        <unicast-configuration>
          <address>192.0.2.1</address>
          <type>uc-server</type>
          <prefer>true</prefer>
          <port>1025</port>
          <authentication>
            <symmetric-key>
              <key-id>10</key-id>
            </symmetric-key>
          </authentication>
        </unicast-configuration>
      </ntp>
    </config>
  </edit-config>
```

An example with IPv6 would used the an IPv6 address (say 2001:db8::1) in the "address" leaf with no change in any other data tree.

```
    <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <target>
        <running/>
      </target>
      <config>
        <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
          <unicast-configuration>
            <address>2001:db8::1</address>
            <type>uc-server</type>
            <prefer>true</prefer>
            <port>1025</port>
            <authentication>
              <symmetric-key>
                <key-id>10</key-id>
              </symmetric-key>
            </authentication>
          </unicast-configuration>
        </ntp>
      </config>
    </edit-config>
```

This example is for retrieving unicast configurations -

```
<get>
  <filter type="subtree">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <unicast-configuration>
      </unicast-configuration>
  </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <unicast-configuration>
      <address>192.0.2.1</address>
      <type>uc-server</type>
        <authentication>
          <symmetric-key>
            <key-id>10</key-id>
          </symmetric-key>
        </authentication>
      <prefer>true</prefer>
      <burst>false</burst>
      <iburst>true</iburst>
      <source/>
      <minpoll>6</minpoll>
      <maxpoll>10</maxpoll>
```

```
        <port>1025</port>
        <stratum>9</stratum>
        <refid>203.0.113.1</refid>
        <reach>255</reach>
        <unreach>0</unreach>
        <poll>128</poll>
        <now>10</now>
        <offset>0.025</offset>
        <delay>0.5</delay>
        <dispersion>0.6</dispersion>
        <originate-time>10-10-2017 07:33:55.253 Z+05:30\
        </originate-time>
        <receive-time>10-10-2017 07:33:55.258 Z+05:30\
        </receive-time>
        <transmit-time>10-10-2017 07:33:55.300 Z+05:30\
        </transmit-time>
        <input-time>10-10-2017 07:33:55.305 Z+05:30\
        </input-time>
        <ntp-statistics>
          <packet-sent>20</packet-sent>
          <packet-sent-fail>0</packet-sent-fail>
          <packet-received>20</packet-received>
          <packet-dropped>0</packet-dropped>
        </ntp-statistics>
      </unicast-configuration>
    </ntp>
  </data>
```

9.2. Refclock master

This example describes how to configure reference clock with stratum 8 -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <refclock-master>
        <master-stratum>8</master-stratum>
      </refclock-master>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get reference clock configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <refclock-master>
      </refclock-master>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <refclock-master>
      <master-stratum>8</master-stratum>
    </refclock-master>
  </ntp>
</data>
```

9.3. Authentication configuration

This example describes how to enable authentication and configure trusted authentication key 10 with mode as AES-CMAC and an hexadecimal string key -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <authentication>
        <auth-enabled>true</auth-enabled>
        <authentication-keys>
          <key-id>10</key-id>
          <algorithm>aes-cmac</algorithm>
          <key>
            <hexadecimal-string>
              bb1d6929e95937287fa37d129b756746
            </hexadecimal-string>
          </key>
          <istrusted>true</istrusted>
        </authentication-keys>
      </authentication>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get authentication related configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <authentication>
      </authentication>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <authentication>
      <auth-enabled>false</auth-enabled>
      <trusted-keys/>
      <authentication-keys>
        <key-id>10</key-id>
        <algorithm>aes-cmac</algorithm>
          <key>
            <hexadecimal-string>
              bb1d6929e95937287fa37d129b756746
            </hexadecimal-string>
          </key>
        <istrusted>true</istrusted>
      </authentication-keys>
    </authentication>
  </ntp>
</data>
```

9.4. Access configuration

This example describes how to configure access mode "peer" associated with acl 2000 -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <access-rules>
        <access-rule>
          <access-mode>peer-access-mode</access-mode>
          <acl>2000</acl>
        </access-rule>
      </access-rules>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get access related configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <access-rules>
      </access-rules>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <access-rules>
      <access-rule>
        <access-mode>peer-access-mode</access-mode>
        <acl>2000</acl>
      </access-rule>
    </access-rules>
  </ntp>
</data>
```

9.5. Multicast configuration

This example describes how to configure multicast-server with address as "224.0.1.1", port as 1025, and version as 3 and authentication keyid as 10 -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <name>Ethernet3/0/0</name>
          <multicast-server>
            <address>224.0.1.1</address>
            <authentication>
              <symmetric-key>
                <key-id>10</key-id>
              </symmetric-key>
            </authentication>
            <port>1025</port>
            <version>3</version>
          </multicast-server>
        </interface>
      </interfaces>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get multicast-server related configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <multicast-server>
          </multicast-server>
        </interface>
      </interfaces>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <interfaces>
      <interface>
        <name>Ethernet3/0/0</name>
        <multicast-server>
          <address>224.0.1.1</address>
          <ttl>8</ttl>
          <authentication>
            <symmetric-key>
              <key-id>10</key-id>
            </symmetric-key>
          </authentication>
          <minpoll>6</minpoll>
          <maxpoll>10</maxpoll>
          <port>1025</port>
          <version>3</version>
        </multicast-server>
      </interface>
    </interfaces>
  </ntp>
</data>
```

This example describes how to configure multicast-client with address as "224.0.1.1" -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <name>Ethernet3/0/0</name>
          <multicast-client>
            <address>224.0.1.1</address>
          </multicast-client>
        </interface>
      </interfaces>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get multicast-client related configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <multicast-client>
          </multicast-client>
        </interface>
      </interfaces>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <interfaces>
      <interface>
        <name>Ethernet3/0/0</name>
        <multicast-client>
          <address>224.0.1.1</address>
        </multicast-client>
      </interface>
    </interfaces>
  </ntp>
</data>
```

9.6. Manycast configuration

This example describes how to configure manycast-client with address as "224.0.1.1", port as 1025 and authentication keyid as 10 -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <name>Ethernet3/0/0</name>
          <manycast-client>
            <address>224.0.1.1</address>
            <authentication>
              <symmetric-key>
                <key-id>10</key-id>
              </symmetric-key>
            </authentication>
            <port>1025</port>
          </manycast-client>
        </interface>
      </interfaces>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get manycast-client related configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <manycast-client>
          </manycast-client>
        </interface>
      </interfaces>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <interfaces>
      <interface>
        <name>Ethernet3/0/0</name>
        <manycast-client>
          <address>224.0.1.1</address>
          <authentication>
            <symmetric-key>
              <key-id>10</key-id>
            </symmetric-key>
          </authentication>
          <ttl>8</ttl>
          <minclock>3</minclock>
          <maxclock>10</maxclock>
          <beacon>6</beacon>
          <minpoll>6</minpoll>
          <maxpoll>10</maxpoll>
          <port>1025</port>
        </manycast-client>
      </interface>
    </interfaces>
  </ntp>
</data>
```

This example describes how to configure manycast-server with address as "224.0.1.1" -

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <config>
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <name>Ethernet3/0/0</name>
          <manycast-server>
            <address>224.0.1.1</address>
          </manycast-server>
        </interface>
      </interfaces>
    </ntp>
  </config>
</edit-config>
```

This example describes how to get manycast-server related configuration -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <interfaces>
        <interface>
          <manycast-server>
          </manycast-server>
        </interface>
      </interfaces>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <interfaces>
      <interface>
        <name>Ethernet3/0/0</name>
        <manycast-server>
          <address>224.0.1.1</address>
        </manycast-server>
      </interface>
    </interfaces>
  </ntp>
</data>
```

9.7. Clock state

This example describes how to get clock current state -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <clock-state>
      </clock-state>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <clock-state>
      <system-status>
        <clock-state>synchronized</clock-state>
        <clock-stratum>7</clock-stratum>
        <clock-refid>192.0.2.1</clock-refid>
        <associations-address>192.0.2.1\
        </associations-address>
        <associations-local-mode>client\
        </associations-local-mode>
        <associations-isconfigured>yes\
        </associations-isconfigured>
        <nominal-freq>100.0</nominal-freq>
        <actual-freq>100.0</actual-freq>
        <clock-precision>18</clock-precision>
        <clock-offset>0.025</clock-offset>
        <root-delay>0.5</root-delay>
        <root-dispersion>0.8</root-dispersion>
        <reference-time>10-10-2017 07:33:55.258 Z+05:30\
        </reference-time>
        <sync-state>clock-synchronized</sync-state>
      </system-status>
    </clock-state>
  </ntp>
</data>
```

9.8. Get all association

This example describes how to get all association present in the system -

```
   <get>
     <filter type="subtree">
       <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
         <associations>
         </associations>
       </ntp>
     </filter>
   </get>

   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
       <associations>
         <address>192.0.2.1</address>
         <stratum>9</stratum>
         <refid>203.0.113.1</refid>
         <local-mode>client</local-mode>
         <isconfigured>true</isconfigured>
         <authentication-key>10</authentication-key>
         <prefer>true</prefer>
         <peer-interface>Ethernet3/0/0</peer-interface>
         <minpoll>6</minpoll>
         <maxpoll>10</maxpoll>
         <port>1025</port>
         <version>4</version>
         <reach>255</reach>
         <unreach>0</unreach>
         <poll>128</poll>
         <now>10</now>
         <offset>0.025</offset>
         <delay>0.5</delay>
         <dispersion>0.6</dispersion>
         <originate-time>10-10-2017 07:33:55.253 Z+05:30\
         </originate-time>
         <receive-time>10-10-2017 07:33:55.258 Z+05:30\
         </receive-time>
         <transmit-time>10-10-2017 07:33:55.300 Z+05:30\
         </transmit-time>
         <input-time>10-10-2017 07:33:55.305 Z+05:30\
         </input-time>
         <ntp-statistics>
           <packet-sent>20</packet-sent>
           <packet-sent-fail>0</packet-sent-fail>
           <packet-received>20</packet-received>
           <packet-dropped>0</packet-dropped>
         </ntp-statistics>
       </associations>
     </ntp>
   </data>
```

9.9. Global statistic

This example describes how to get global statistics -

```
<get>
  <filter type="subtree">
    <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
      <ntp-statistics>
      </ntp-statistics>
    </ntp>
  </filter>
</get>

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ntp xmlns="urn:ietf:params:xml:ns:yang:ietf-ntp">
    <ntp-statistics>
      <packet-sent>30</packet-sent>
      <packet-sent-fail>5</packet-sent-fail>
      <packet-received>20</packet-received>
      <packet-dropped>2</packet-dropped>
    </ntp-statistics>
  </ntp>
</data>
```

10. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

URI: urn:ietf:params:xml:ns:yang:ietf-ntp

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

This document registers a YANG module in the "YANG Module Names" registry [RFC6020].

Name: ietf-ntp

Namespace: urn:ietf:params:xml:ns:yang:ietf-ntp

Prefix: ntp

Reference: RFC XXXX

Note: The RFC Editor will replace XXXX with the number assigned to this document once it becomes an RFC.

11. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/ntp/port - This data node specify the port number to be used to send NTP packets. Unexpected changes could lead to disruption and/or network misbehavior.

/ntp/authentication and /ntp/access-rules - The entries in the list include the authentication and access control configurations. Care should be taken while setting these parameters.

/ntp/unicast-configuration - The entries in the list include all unicast configurations (server or peer mode), and indirectly creates or modify the NTP associations. Unexpected changes could lead to disruption and/or network misbehavior.

/ntp/interfaces/interface - The entries in the list include all per-interface configurations related to broadcast, multicast and manycast mode, and indirectly creates or modify the NTP associations. Unexpected changes could lead to disruption and/or network misbehavior.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or

notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/ntp/authentication/authentication-keys - The entries in the list includes all the NTP authentication keys. This information is sensitive and can be exploited and thus unauthorized access to this needs to be curtailed.

/ntp/associations - The entries in the list includes all active NTP associations of all modes. Unauthorized access to this also needs to be curtailed.

The leaf /ntp/authentication/authentication-keys/algorithm can be set to cryptographic algorithms that are no longer considered to be secure. As per [RFC8573], AES-CMAC is the recommended algorithm.

12. Acknowledgments

The authors would like to express their thanks to Sladjana Zoric, Danny Mayer, Harlan Stenn, Ulrich Windl, Miroslav Lichvar, Maurice Angermann, Watson Ladd, and Rich Salz for their review and suggestions.

Thanks to Andy Bierman for the YANG doctor review.

Thanks to Dieter Sibold for being the document shepherd and Erik Kline for being the responsible AD.

Thanks to Takeshi Takahashi for SECDIR review. Thanks to Tim Evens for GENART review.

A special thanks to Tom Petch for a very detailed YANG review and providing great suggestions for improvements.

13. References

13.1. Normative References

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[RFC5907] Gerstung, H., Elliott, C., and B. Haberman, Ed., "Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)", RFC 5907, DOI 10.17487/RFC5907, June 2010, <https://www.rfc-editor.org/info/rfc5907>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[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>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

[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>.

13.2. Informative References

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, DOI 10.17487/RFC1305, March 1992, <https://www.rfc-editor.org/info/rfc1305>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[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>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

Appendix A. Full YANG Tree

The full tree for ietf-ntp YANG model is -

```
module: ietf-ntp
  +--rw ntp!
     +--rw port?                    inet:port-number {ntp-port}?
     +--rw refclock-master!
     |  +--rw master-stratum?   ntp-stratum
     +--rw authentication {authentication}?
     |  +--rw auth-enabled?          boolean
     |  +--rw authentication-keys* [key-id]
     |     +--rw key-id       uint32
     |     +--rw algorithm?   identityref
     |     +--rw key
     |     |  +--rw (key-string-style)?
     |     |     +--:(keystring)
     |     |     |  +--rw keystring?            string
     |     |     +--:(hexadecimal) {hex-key-string}?
     |     |        +--rw hexadecimal-string?   yang:hex-string
     |     +--rw istrusted?   boolean
     +--rw access-rules {access-rules}?
     |  +--rw access-rule* [access-mode]
     |     +--rw access-mode    identityref
     |     +--rw acl?           -> /acl:acls/acl/name
     +--ro clock-state
     |  +--ro system-status
     |     +--ro clock-state                 identityref
     |     +--ro clock-stratum               ntp-stratum
     |     +--ro clock-refid                 refid
```

```
|      +--ro associations-address?
|      |       -> /ntp/associations/address
|      +--ro associations-local-mode?
|      |       -> /ntp/associations/local-mode
|      +--ro associations-isconfigured?
|      |       -> /ntp/associations/isconfigured
|      +--ro nominal-freq                 decimal64
|      +--ro actual-freq                  decimal64
|      +--ro clock-precision              int8
|      +--ro clock-offset?                decimal64
|      +--ro root-delay?                  decimal64
|      +--ro root-dispersion?             decimal64
|      +--ro reference-time?              ntp-date-and-time
|      +--ro sync-state                   identityref
|    +--rw unicast-configuration* [address type]
|    |       {unicast-configuration}?
|    |  +--rw address           inet:ip-address
|    |  +--rw type              identityref
|    |  +--rw authentication {authentication}?
|    |  |  +--rw (authentication-type)?
|    |  |     +--:(symmetric-key)
|    |  |        +--rw key-id?   leafref
|    |  +--rw prefer?           boolean
|    |  +--rw burst?            boolean
|    |  +--rw iburst?           boolean
|    |  +--rw source?           if:interface-ref
|    |  +--rw minpoll?          int8
|    |  +--rw maxpoll?          int8
|    |  +--rw port?             inet:port-number {ntp-port}?
|    |  +--rw version?          ntp-version
|    +--ro associations* [address local-mode isconfigured]
|    |  +--ro address           inet:ip-address
|    |  +--ro local-mode        identityref
|    |  +--ro isconfigured      boolean
|    |  +--ro stratum?          ntp-stratum
|    |  +--ro refid?            refid
|    |  +--ro authentication?
|    |  |       -> /ntp/authentication/authentication-keys/key-id
|    |  |       {authentication}?
|    |  +--ro prefer?           boolean
|    |  +--ro peer-interface?   if:interface-ref
|    |  +--ro minpoll?          int8
|    |  +--ro maxpoll?          int8
|    |  +--ro port?             inet:port-number {ntp-port}?
|    |  +--ro version?          ntp-version
|    |  +--ro reach?            uint8
|    |  +--ro unreach?          uint8
|    |  +--ro poll?             int8
```

```
|  +--ro now?              uint32
|  +--ro offset?           decimal64
|  +--ro delay?            decimal64
|  +--ro dispersion?       decimal64
|  +--ro originate-time?   ntp-date-and-time
|  +--ro receive-time?     ntp-date-and-time
|  +--ro transmit-time?    ntp-date-and-time
|  +--ro input-time?       ntp-date-and-time
|  +--ro ntp-statistics
|     +--ro discontinuity-time?   ntp-date-and-time
|     +--ro packet-sent?          yang:counter32
|     +--ro packet-sent-fail?     yang:counter32
|     +--ro packet-received?      yang:counter32
|     +--ro packet-dropped?       yang:counter32
+--rw interfaces
|  +--rw interface* [name]
|     +--rw name                if:interface-ref
|     +--rw broadcast-server! {broadcast-server}?
|     |  +--rw ttl?              uint8
|     |  +--rw authentication {authentication}?
|     |  |  +--rw (authentication-type)?
|     |  |     +--:(symmetric-key)
|     |  |        +--rw key-id?   leafref
|     |  +--rw minpoll?          int8
|     |  +--rw maxpoll?          int8
|     |  +--rw port?             inet:port-number {ntp-port}?
|     |  +--rw version?          ntp-version
|     +--rw broadcast-client! {broadcast-client}?
|     +--rw multicast-server* [address] {multicast-server}?
|     |  +--rw address
|     |  |       rt-types:ip-multicast-group-address
|     |  +--rw ttl?              uint8
|     |  +--rw authentication {authentication}?
|     |  |  +--rw (authentication-type)?
|     |  |     +--:(symmetric-key)
|     |  |        +--rw key-id?   leafref
|     |  +--rw minpoll?          int8
|     |  +--rw maxpoll?          int8
|     |  +--rw port?             inet:port-number {ntp-port}?
|     |  +--rw version?          ntp-version
|     +--rw multicast-client* [address] {multicast-client}?
|     |  +--rw address    rt-types:ip-multicast-group-address
|     +--rw manycast-server* [address] {manycast-server}?
|     |  +--rw address    rt-types:ip-multicast-group-address
|     +--rw manycast-client* [address] {manycast-client}?
|        +--rw address
|        |       rt-types:ip-multicast-group-address
|        +--rw authentication {authentication}?
```

```
         |         |  +--rw (authentication-type)?
         |         |     +--:(symmetric-key)
         |         |        +--rw key-id?   leafref
         |        +--rw ttl?              uint8
         |        +--rw minclock?         uint8
         |        +--rw maxclock?         uint8
         |        +--rw beacon?           int8
         |        +--rw minpoll?          int8
         |        +--rw maxpoll?          int8
         |        +--rw port?             inet:port-number {ntp-port}?
         |        +--rw version?          ntp-version
        +--ro ntp-statistics
           +--ro discontinuity-time?   ntp-date-and-time
           +--ro packet-sent?          yang:counter32
           +--ro packet-sent-fail?     yang:counter32
           +--ro packet-received?      yang:counter32
           +--ro packet-dropped?       yang:counter32
```

Authors' Addresses

Nan Wu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: eric.wu@huawei.com

Dhruv Dhody (editor)
Huawei
Divyashree Techno Park, Whitefield
Bangalore, Kanataka 560066
India

Email: dhruv.ietf@gmail.com

Ankit kumar Sinha (editor)
RtBrick Inc.
Bangalore, Kanataka
India

Email: ankit.ietf@gmail.com

Anil Kumar S N
RtBrick Inc.
Bangalore, Kanataka
India

Email: anil.ietf@gmail.com

Yi Zhao
Ericsson
China Digital Kingdom Bld., No.1 WangJing North Rd.
Beijing 100102
China

Email: yi.z.zhao@ericsson.com

TICTOC Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 14, 2021

D. Arnold
Meinberg-USA
H. Gerstung
Meinberg

December 11, 2020

Enterprise Profile for the Precision Time Protocol With Mixed Multicast and Unicast Messages
draft-ietf-tictoc-ptp-enterprise-profile-18

Abstract

This document describes a profile for the use of the Precision Time Protocol in an IPV4 or IPv6 Enterprise information system environment. The 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 June 14, 2021.

Table of Contents

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 ("slave clock") in IEEE 1588-2008 [IEEE1588] needs to know the identity of the time sources in the network (the Master Clocks).

The "Best Master Clock Algorithm" (IEEE 1588-2008 [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 shall be the active sending time source (Master Clock). 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 second revision of the standard (IEEE 1588-2008) is the current version (also known as PTPv2) and introduced 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 more than 99% of the received multicast messages because they carried information for some other node. PTPv2 also introduced PTP profiles (IEEE 1588-2008 [IEEE1588] subclause 19.3). 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 profile on a PTP node will set all the parameters that are specified in the profile to a defined value. If a PTP profile definition allows multiple values for a parameter, selection of the 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 profile should also define which optional features of PTP are required, permitted, or prohibited. It is possible to extend the PTP standard with a PTP profile by using the TLV mechanism of PTP (see IEEE 1588-2008 [IEEE1588] subclause 13.4), defining an optional Best Master Clock Algorithm and a few other ways. PTP has its own management protocol (defined in IEEE 1588-2008 [IEEE1588] subclause 15.2) but allows a PTP profile specify an alternative management mechanism, for example SNMP.

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].

3. Technical Terms

- Acceptable Master Table: A PTP Slave Clock may maintain a list of masters which it is willing to synchronize to.

- Alternate Master: A PTP Master Clock, which is not the Best Master, may act as a master with the Alternate Master flag set on the messages it sends.

- Announce message: Contains the Master Clock properties of a Master Clock. Used to determine the Best Master.

- Best Master: A clock with a port in the master state, operating consistently with the Best Master Clock Algorithm.

- Best Master Clock Algorithm: A method for determining which state a port of a PTP clock should be in. The algorithm works by identifying which of several PTP Master capable clocks is the best master. Clocks have priority to become the acting Grandmaster, based on the properties each Master Clock sends in its Announce Message.

- Boundary Clock: A device with more than one PTP port. Generally boundary Clocks will have one port in slave state to receive timing and then other ports in master state to re-distribute the timing.

- Clock Identity: In IEEE 1588-2008 this is a 64-bit number assigned to each PTP clock which must be unique. Often it is derived from the Ethernet MAC address, since there is already an international infrastructure for assigning unique numbers to each device manufactured.

- 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 Master Clock and Slave Clock.

- Grandmaster: the primary Master Clock within a domain of a PTP system

- IEEE 1588: The timing and synchronization standard which defines PTP, and describes the node, system, and communication properties necessary to support PTP.

- Master Clock: a clock with at least one port in the master 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 Master Clock, or be a slave clock.

- Peer to Peer Delay Measurement Mechanism: A network delay measurement mechanism in PTP facilitated by an exchange of messages between adjacent devices in a network.

- Preferred Master: 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, a unicast slave which talks to several Grandmaster clocks in parallel.

- PTPv2: Refers specifically to the second version of PTP defined by IEEE 1588-2008.

- Rogue Master: A clock with a port in the master state, even though it should not be in the master state according to the Best Master Clock Algorithm, and does not set the alternate master flag.

- Slave clock: a clock with at least one port in the slave state, and no ports in the master state.

- Slave Only Clock: An Ordinary Clock which cannot become a Master 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 slaves to establish a unicast communication with PTP masters using a configures table of master IP addresses and Unicast Message Negotiation.

- Unicast Negotiation: A mechanism in PTP for Slave Clocks to negotiate unicast Sync, announce and Delay Request Message Rates from a Master Clock.

4. Problem Statement

This document describes a version of PTP intended to work in large enterprise networks. Such 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 100 nanoseconds to the Grandmaster. This 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 hardware time stamping, and on path support. These features are currently part of PTP, or are allowed by it. Because PTP has a complex range of features and options it is necessary to create a 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 unique to one device as a unicast message. The latter can be essential as the number of PTP slaves 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 SHALL 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 D and E in IEEE 1588 [IEEE1588] respectively. If a network contains both IPv4 and IPv6, then they SHALL be treated as separate communication paths. Clocks which communicate using IPv4 can interact with clocks using IPv6 if there is an intermediary device which simultaneously communicates with both IP versions. A Boundary Clock might perform this function, for example. A PTP domain SHALL use either IPv4 or IPv6 over a communication path, but not both. 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 Masters, Ordinary Clocks, or Boundary Clocks. The Ordinary Clocks may be Slave Only Clocks, or be master capable.

Note that clocks SHOULD always be identified by their clock ID and not the IP or Layer 2 address. This is important in IPv6 networks

since Transparent Clocks are required to change the source address of any packet which they alter. 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, like 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 Master to the Slave. 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

Master Clocks, Transparent Clocks and Boundary Clocks MAY be either one-step clocks or two-step clocks. Slave 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 master and slave do not necessarily traverse the same physical path. Thus, wherever possible, the network SHOULD be traffic 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 SHALL send Follow-up messages as PTP general 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:0:0:0:0:0:0:181 for Ipv6, where X can be a value between 0x0 and 0xF, see IEEE 1588 [IEEE1588] Annex E, Section E.3.

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

Clocks SHOULD include support for multiple domains. The purpose is to support multiple simultaneous masters for redundancy. Leaf devices (non-forwarding devices) can use timing information from

multiple masters by combining information from multiple instantiations of a PTP stack, each operating in a different domain. Redundant sources of timing can be ensembled, and/or compared to check for faulty Master Clocks. The use of multiple simultaneous masters will help mitigate faulty masters reporting as healthy, network delay asymmetry, and security problems. Security problems include man-in-the-middle attacks such as delay attacks, packet interception / manipulation attacks. Assuming the path to each master 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 SHALL 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 SHALL NOT be changed from the default value. The Announce Receipt Timeout Interval SHALL be three Announce Intervals for Preferred Masters, and four Announce Intervals for all other masters.

The logMessageInterval carried in the unicast Delay Response message MAY be set to correspond to the master 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 Master Clocks

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

A port of a clock SHALL NOT be in the master state unless the clock has a current value for the number of UTC leap seconds.

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

9. Requirements for Slave Clocks

Slave clocks MUST be able to operate properly in a network which contains multiple Masters in multiple domains. Slaves SHOULD make use of information from the all Masters in their clock control

subsystems. Slave Clocks MUST be able to operate properly in the presence of a Rogue Master. Slaves SHOULD NOT Synchronize to a Master which is not the Best Master in its domain. Slaves will continue to recognize a Best Master for the duration of the Announce Time Out Interval. Slaves MAY use an Acceptable Master Table. If a Master is not an Acceptable Master, then the Slave MUST NOT synchronize to it. Note that IEEE 1588-2008 requires slave clocks to support both two-step or one-step Master clocks. See IEEE 1588 [IEEE1588], subClause 11.2.

Since Announce messages are sent as multicast messages slaves can obtain the IP addresses of a master from the Announce messages. Note that the IP source addresses of Sync and Follow-up messages may have been replaced by the source addresses of a Transparent Clock, so, slaves 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 profile.

10. Requirements for Transparent Clocks

Transparent Clocks SHALL NOT change the transmission mode of an Enterprise Profile PTP message. For example, a Transparent Clock SHALL NOT change a unicast message to a multicast message. Transparent Clocks SHOULD support multiple domains. Transparent Clocks which syntonize to the master clock will 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 servo loops 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 for a specific clock.

13. Forbidden PTP Options

Clocks operating in the Enterprise Profile SHALL NOT use peer to peer timing for delay measurement. Grandmaster Clusters are NOT ALLOWED. The Alternate Master option is also NOT ALLOWED. Clocks operating in the Enterprise Profile SHALL NOT use Alternate Timescales. Unicast discovery and unicast negotiation SHALL NOT be used.

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 J.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 J.3 Default Profile Clocks. The Enterprise Profile will not interoperate with the Annex J.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 profiles if the clocks in the other profiles obey the rules of the Enterprise Profile. These rules MUST NOT be changed to achieve interoperability with other profiles.

15. Profile Identification

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

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

This 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 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 masters, using multiple PTP domains can mitigate problems from rogue masters and man-in-the-middle attacks. 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 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. 1588-2008, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems."", 7 2008, <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>.

[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

[G8271] International Telecommunication Union, "ITU-T G.8271/ Y.1366, "Time and Phase Synchronization Aspects of Packet Networks"", 2 2012, <https://www.itu.int>.

[ISPCS] Arnold, D., "Plugfest Report", 10 2017, <https://www.ispcs.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>.

[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
USA

Email: doug.arnold@meinberg-usa.com

Heiko Gerstung
Meinberg
Lange Wand 9
Bad Pyrmont 31812
Germany

Email: heiko.gerstung@meinberg.de

Network Time Protocol
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2021

M. Langer
R. Bermbach
Ostfalia University

March 8, 2021

NTS4PTP - Key Management System for the Precision Time Protocol Based on the Network Time Security Protocol
draft-langer-ntp-nts-for-ptp-01

Abstract

This document defines a key management service for automatic key management for the integrated security mechanism (Prong A) of IEEE Std 1588[TM]-2019 described there in Annex P. It implements a key management for immediate security processing complementing the exemplary GDOI proposal in P.2.1.2.1. The key management service is based on the "NTS Key Establishment" protocol defined in IETF RFC 8915 for securing NTP, but works completely independent from 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 September 9, 2021.

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. 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.

2. Key Management Using Network Time Security

Many networks include both PTP and NTP at the same time. Furthermore, many time server appliances that are capable of acting as the Grandmaster of a PTP Network are also capable of acting as an NTP server. For these reasons it is likely to be easier both for the time server manufacturer 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. For example, HTTPS, the predominant secure web protocol uses TLS for security. Since many PTP capable network appliances have management interfaces based on HTTPS, the manufacturers are already implementing TLS. This document outlines how the NTS Key Establishment protocol of IETF RFC 8915 can be expanded for use as a PTP key management mechanism [Langer_et_al._2020] for immediate security processing complementing the exemplary GDOI proposal in the IEEE Std 1588-2019

[IEEE1588-2019]. As a key establishment server for NTP should be implemented stateless which is not necessary for PTP systems, suitable new NTS messages are to be defined in this document.

Though the key management for PTP is based on the NTS Key Establishment 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 and expands it with new NTS messages for PTP. It may be applied in a Key Establishment server (KE server) that already manages NTP but can also be operated only handling KE 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 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 KE server program and the PTP instance program needs to be implemented in a secure way if TLS communication (e.g. via local host) is not or cannot be used.

Using the expanded NTS Key Establishment protocol for the NTS key management for PTP, NTS4PTP provides two principle approaches specified in this document.

1. Group-based approach:

- Definition of one or more security groups in the PTP network,
- very suitable for PTP multicast mode and mixed multicast/unicast mode,
- suitable for unicast mode in small subgroups of very few participants (Group-of-2, Go2) but poor scaling and more administration work,

2. Ticket-based approach

- secured (end-to-end) PTP unicast communication between requester and grantor,
- no group binding necessary,
- very suitable for native PTP unicast mode, because of good scaling,
- a bit more complex NTS message handling.

This document describes the structure and usage of these two approaches in their application as a key management system for the integrated security mechanism (Prong A) of IEEE Std 1588-2019. Section 2.1 starts with a description of the principle key distribution mechanism, continues with details of the various group-based options (Section 2.1.1) and the ticket-based unicast mode

(Section 2.1.2) before it ends with more general topics in Section 2.2 for example the key update process and finally an overview of the newly defined NTS messages in Section 2.3. Section 3 gives all the details necessary to construct all records forming the particular NTS messages. Section 4 depicts details of a TICKET TLV needed to transport encrypted security information in PTP unicast requests. The following Section 5 mentions specific parameters used in the PTP AUTHENTICATION TLV when working with the NTS4PTP key management system. Section 6 and Section 7 discuss IANA respectively security considerations.

2.1. Principle Key Distribution Mechanism

A PTP instance requests a key from the server referred to as the Key Establishment server, or (NTS-) KE server. Figure 1 describes the principle sequence which can be used for PTP multicast as well as PTP unicast operation.

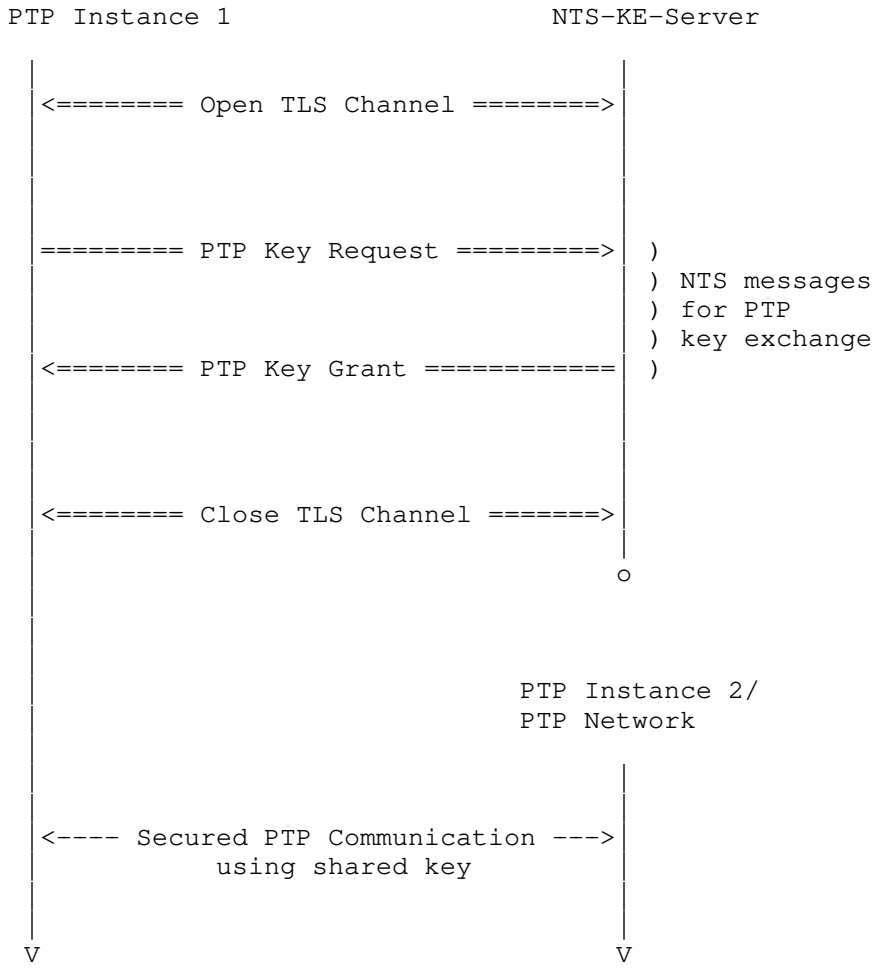


Figure 1: NTS Key distribution sequence

The client connects to the 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. No earlier TLS versions are allowed. The details of the TLS handshake are specified in IETF RFC 8446 [RFC8446].

Implementations must conform to the rules stated in chapter 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 TLS handshake accomplishes the following:

- Negotiation of TLS version (only TLS 1.3 allowed), and
- negotiation of the cipher suite for the TLS session, and
- authentication of the TLS server (equivalent to the KE server) using a digital X.509 certificate,
- verification of the TLS client (PTP instance) using its digital X.509 certificate and
- the encryption of the subsequent information exchange between the TLS communication partners.

TLS therefore enables peer authentication by certificates and provides authenticity, message integrity and confidentiality of following data transmitted over the TLS channel.

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).

Once the TLS session is established, the PTP instance will ask for a PTP key as well as the associated security parameters using the new NTS message PTP Key Request (see Section 2.3.1). The NTS application of the KE server will respond with either a PTP Key Grant message (see Section 2.3.2), or a PTP Refusal message (see Section 2.3.3). All messages are constructed from specific records as described in Section 3.2.

When the Key Request message was responded with a PTP Key Grant or a PTP Refusal the TLS session will be closed with a close notify TLS message from both parties, the PTP instance and the key server.

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 can resume the TLS session by including a TLS session ID, allowing the PTP instance to skip the TLS version and algorithm negotiations. If resuming is used, a suitable lifetime for the TLS session key must be defined to not open the TLS connection for security threats.

As the TLS session provides authentication, but not authorization additional means has to be used for the latter (see Section 2.2.5.4).

As mentioned above, the NTS key management for PTP supports two principle methods, the group-based approach and the ticket-based approach which are described in the following sections below.

2.1.1. NTS Message Exchange for Group-based Approach

As described in Section 2.1, a PTP instance wanting to join a secured PTP communication in the group-based modes contacts the KE server inside a secured TLS connection with a PTP Key Request message (see Section 2.3.1) as shown in Figure 2. The KE server answers with a PTP Key Grant message (see Section 2.3.2) with all the necessary data to join the group communication or with a PTP Refusal message (see Section 2.3.3) 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 KE server. The group number for both modes is primarily generated by a concatenation of the PTP domain number and the PTP profile (sdoId), as described in Section 3.2.2.

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.

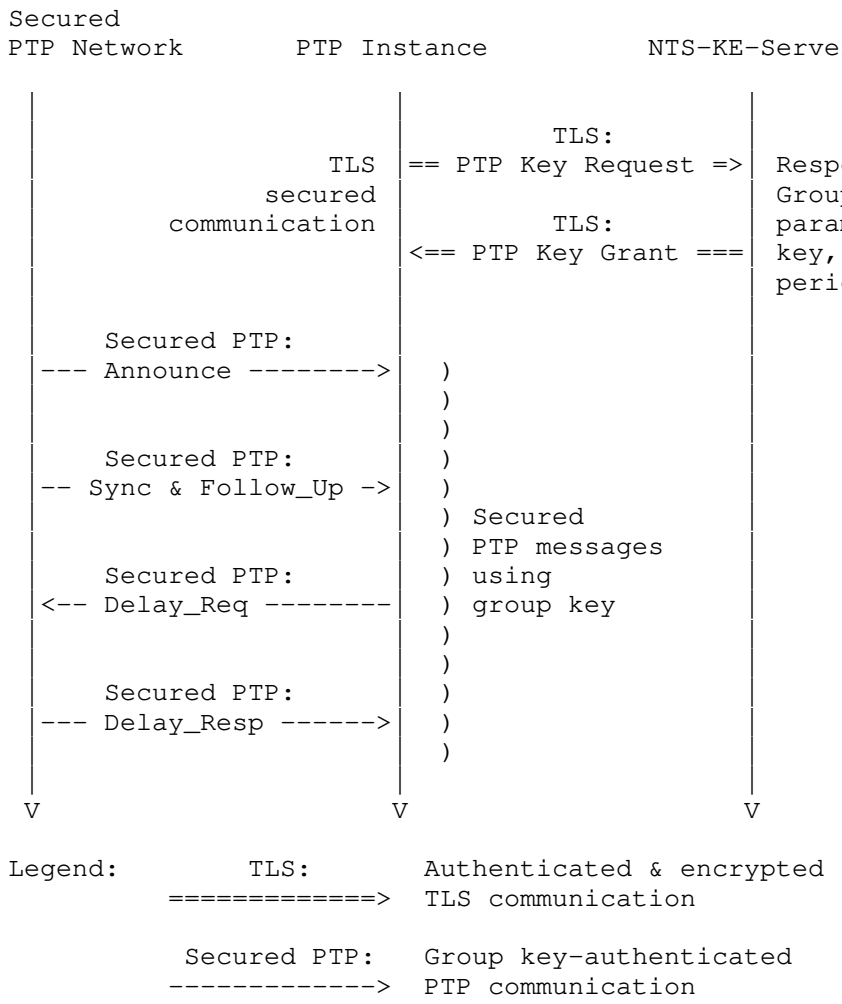


Figure 2: Message exchange for the group-based approach

This 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 3.2.2) 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 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 2) 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 key management for these modes works relatively simple and needs only the above mentioned three NTS messages: PTP Key Request, PTP Key Grant or PTP Refusal. The group number used for addressing is automatically derived from the configured attributes domain number and sdoID.

Additionally, besides multicast and hybrid mode, a (multicast) group of two PTP instances can be configured, practically implementing a special unicast communication.

The key management for these modes works relatively simple and needs only the above mentioned three NTS messages: PTP Key Request, PTP Key Grant or PTP Refusal. The group number used for addressing is automatically derived from the configured attributes PTP domain number and sdoId. For Go2, the attribute subGroup is additionally required.

2.1.2. NTS Message Exchange for the Ticket-based Approach

In (native) PTP unicast mode using unicast message negotiation ([IEEE1588-2019], 16.1) any potential instance (the grantor) which can be contacted by other PTP instances (the requesters) needs to register upfront with the KE server as depicted in Figure 3.

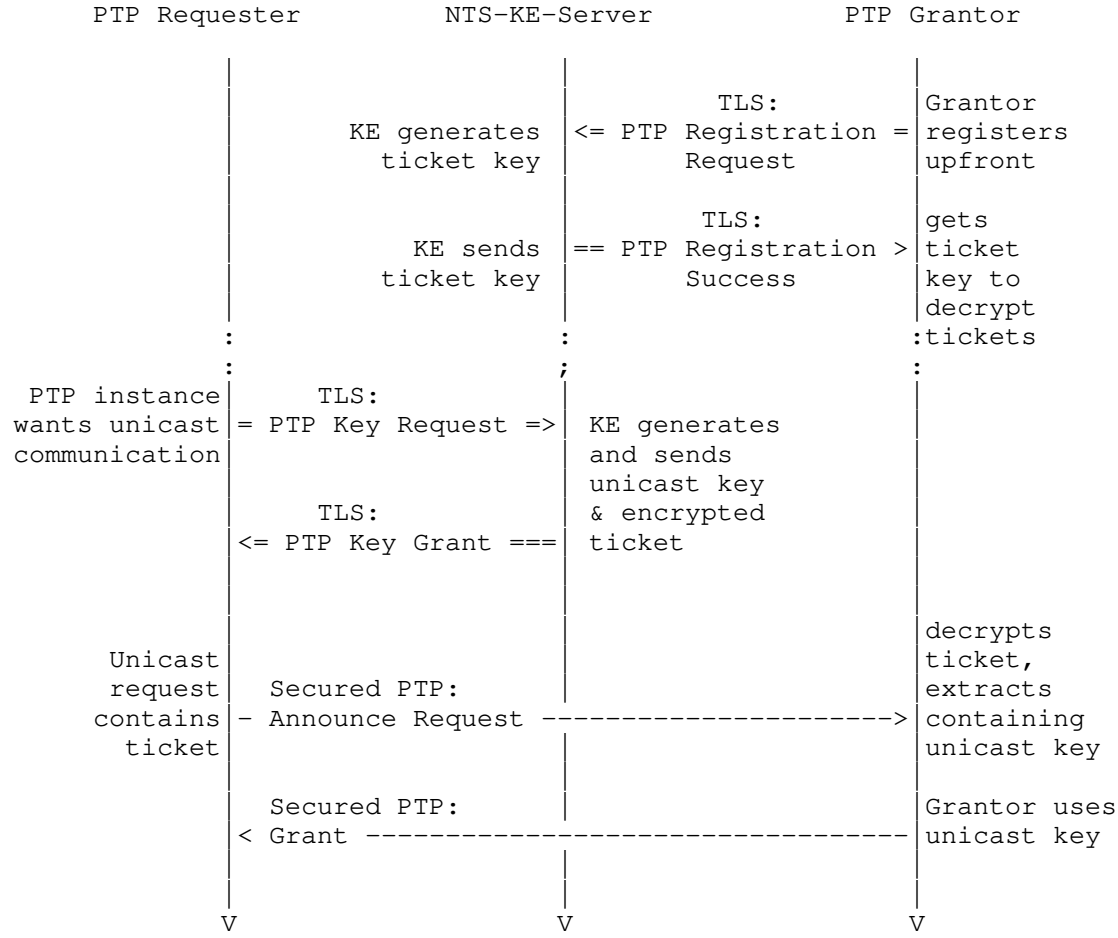


Figure 3: Message exchange for ticket-based unicast mode

(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 resp. 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.)

This registration is performed via a PTP Registration Request message (see Section 2.3.4). The KE server answers with a PTP Registration Success message (see Section 2.3.5) or a PTP Refusal message (see Section 2.3.3).

With the reception of the PTP Registration Success message the grantor holds a ticket key known only to the 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.

As with the group-based approach, a PTP instance (the requester) wanting to start a secured PTP unicast communication with a specific grantor contacts the KE server sending a PTP Key Request message (see Section 2.3.1) as shown in Figure 3 using the TLS-secured NTS Key Establishment protocol. The KE server answers with a PTP Key Grant message (see Section 2.3.2) with all the necessary data to begin the unicast communication with the desired partner or with a PTP Refusal message (see Section 2.3.3) if unicast communication with that instance is unavailable.

The PTP Key Grant message includes a unicast key to secure the PTP message exchange with the desired grantor. In addition, it contains the above mentioned encrypted ticket which the requester transmits in a special Ticket TLV (see Section 4) with the secured PTP message to the grantor. 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 requester’s 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.

After the NTS key establishment messages for the PTP unicast mode have been exchanged the secured PTP communication can take place using the Security Association(s) communicated.

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 2.3.6) to the KE server, so requesters no longer receive PTP Key Grant messages for this grantor.

This unicast mode is a bit more complex than the Group-of-2 approach 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.

2.2. General Topics

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

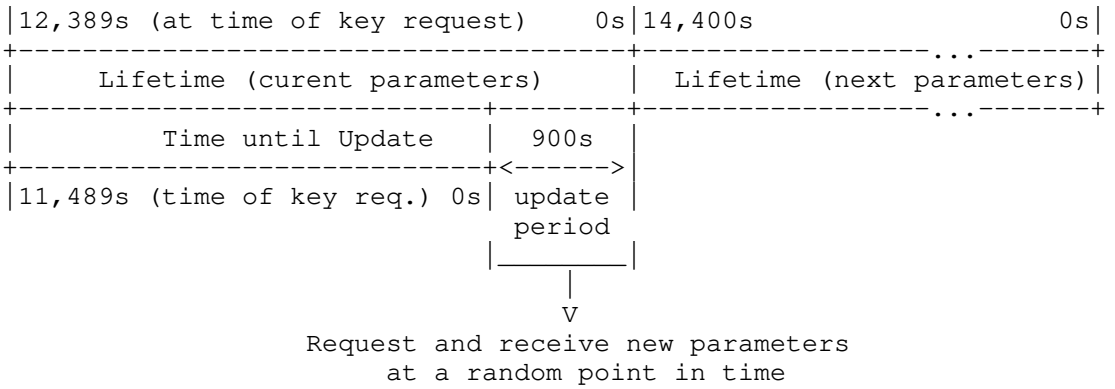
2.2.1. Key Update Process

All keys are equipped with parameters for a specific lifetime. Thereafter new key material has to be used. The value in the Lifetime record given by the KE server in the respective NTS messages is specified in seconds which denote the remaining time until the key expires and are decremented down to zero. So hard adjustments of the clock used have to be avoided. Therefore the use of a monotonic clock is recommended. Requests during the currently running lifetime will receive respectively adapted count values.

The receiving instances may concede a Grace Time in the range of, for example 5 - 10 seconds where an old key is still accepted to handle internal delays gracefully. The Grace Time may be defined in a PTP profile. Additionally, the KE server can optionally be configured to inform about a grace time value generally to be used.

New security parameters will be available after the Time until Update (TuU). The Time until Update given by the KE server is specified in seconds which are decremented down to zero. After that point in time until the end of the Lifetime of an associated key the PTP instances should connect to the KE server again, to receive new security parameters. The actual point in time, when a PTP instance asks for new data, should be selected randomly in the update period - the time after TuU was decremented to zero and before the Lifetime is counted down completely - to avoid peak load on the KE server. Figure 4 presents an example of the key update mechanism. A PTP instance sending a PTP Key Request to the KE server during the update period will receive the current security parameters (Current Parameters) as well as the security parameters of the following period (Next Parameters). As with the lifetime, requests during the currently running lifetime will receive respectively adapted count values for the current TuU.

Lifetime and Time until Update allow a cyclic rotation of security parameters during the running operation. This approach guarantees continuous secured PTP communication without interruption by key rotation.



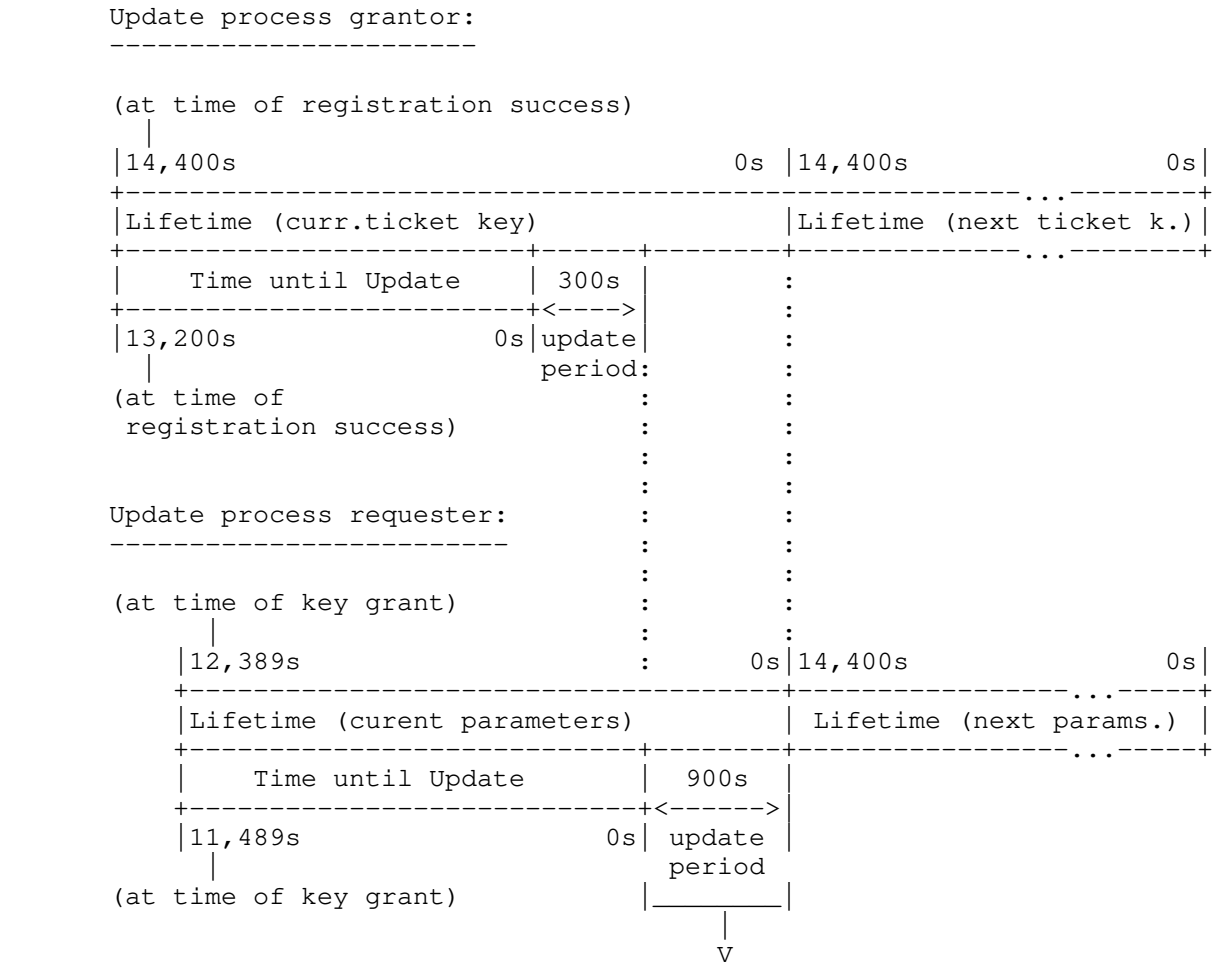
Example:

Lifetime (full): 14,400s = 4h
Time unitil Update (full): 13,500s -> updated period: 900s = 15 min

Figure 4: Example of the parameter rotation using Lifetime and Time until Update in group-based mode

The key rotation mechanism described also applies for the ticket-based approach. As there are two keys, the ticket key and the unicast key, some details need to be explained (see Figure 5). When the grantor registers with the KE server it receives the ticket key with the PTP Registration Success message together with the Lifetime and the respective Time until Update records. The lifetime parameters also apply to the ticket a requester would receive.

A requester wanting to communicate in unicast sends a PTP Key Request message with the particular parameters to the KE server. In the response it receives a specific unicast key with Lifetime and TuU as well as the encrypted ticket containing all the necessary security information for the grantor. The lifetime of the unicast key will end at the same point in time as the ticket key. Requests during the currently running lifetime of the ticket key will receive respectively adapted count values. The lifetime can be at most the remaining lifetime of the respective ticket key of the grantor.



Example:

Lifetime (full): 14,400s = 4h
Time unitil Update (full):
- requester 13,500s -> updated period: 900s = 15 min
Time unitil Update (full):
- grantor: 13,200s = ToU of requester - 300s

Figure 5: Example of the parameter rotation using Lifetime and Time until Update in ticket-based mode

The TuU of the ticket key will end earlier than the TuU of associated unicast keys. The grantor should re-register in its update period beginning after the Time until Update of the ticket key was decremented to zero and ending when an associated unicast key TuU is counted down. As the grantor does not know how long its update period lasts it should re-register immediately after its TuU has ended. (A profile or a general configuration may fix the length of a grantors' update period. Then the grantor could re-register at a random point in time during its update period. Because masters register asynchronously, their re-registration will also be asynchronous. So typically, no peak load for the KE server will be generated.) Its update period is a mere timing buffer for cases where re-registration will not work instantly. The re-registration should be completed before any requester can start a PTP Key Request for ticket-based unicast mode. This guarantees the availability of a new ticket. When re-registering in its update period the grantor will receive together with the ticket key, etc., Lifetime and Time until Update of the current period as well as the parameters of the following period - similar to multicast keys. (A registration during the TuU period will supply only current data, not parameters of the following period. A late re-registration after the end of the current Lifetime will start a new period with respective full lifetime und update parameters.)

A requester needs to ask for a new unicast key and ticket at the KE server during the update period for uninterrupted unicast communication possibility or else at any later point in time. During the update period it will receive the Current Parameters as well as the Next Parameters. Embedded in the respective data, it will receive the ticket for the grantor including the encrypted ticket. Each ticket carries the same security information as the respective Current Parameters or Next Parameters data structure.

If a grantor does not have re-registered (in time or at all) when corresponding requesters try to get unicast keys, they will receive a PTP Refusal message.

If a grantor has revoked his registration with a PTP Registration Revoke message, requesters will receive a PTP Refusal message when trying to update for a new unicast key. No immediate key revoke mechanism exists. The grantor should not grant respective unicast requests until the revoked key expires.

2.2.2. Key Generation

In all cases keys obtained by a secure random number generator shall be used. The length of the keys depends on the MAC algorithm (see

also last subsection in Section 3.3.2) respectively the AEAD algorithm utilized.

2.2.3. Time Information of the KE Server

As the KE server embeds time duration information in the respective messages, its local time should be sufficiently precise to a maximum 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.

2.2.4. 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 2.2.5.4.) 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], chapter 8.5 "Initial Verification of Server certificates" which applies to client 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) 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 KE server would be necessary: which domain, sdoId and/or subgroup attributes belong to which certificate.

As TLS is used to secure the NTS Key Establishment 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.

2.2.5. 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.

2.2.5.1. 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 KE server. This includes the Security Policies, i.e. which PTP messages are to be secured. 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 KE server and PTP instances to implement an NTS algorithm negotiation protocol.

For the ticket-based approach the AEAD algorithms need to be specified which the PTP grantors and the 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 protocol supported by every instance should be defined. Eventually, suitable algorithms may be defined in a respective profile.

2.2.5.2. Key Lifetimes

Supplementary to the above mentioned SAs the desired key rotation periods, i.e. the lifetimes of keys resp. 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 approach as well as the lifetime of ticket key and unicast key in the ticket-based unicast approach (typically for every unicast pair in general or eventually specific for each requestor-grantor pair). In addition, the corresponding Time until Update parameters need to be defined which (together with the lifetime) specify the relevant update period. Any particular Lifetime and Time until Update are configured as time spans counted in seconds and start at the same point in time.

2.2.5.3. Certificates

The network administrator has to supply each PTP instance and the KE server with their X.509 certificates. The TLS communication parties must be equipped with a private key and a certificate containing the public key in advance (see Section 2.2.4).

2.2.5.4. 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 approach 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 2.2.4.)

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 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 member of the group used for multicast may request a registration for unicast

communication at the KE server. If it is intended for unicast, this must be configured locally. If no group authorization is available (e.g. pure unicast operation) another authentication scheme is necessary.

In the same way, any requester (if configured for it locally) may request security data for a unicast connection with a specific grantor. Only authentication at the KE server using its certificate and membership in the group used for multicast is needed. 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. pure unicast 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 lies out of scope of this document.

2.2.5.5. 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 KE server to allow secure use of the correctionField 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 approaches multicast mode and mixed multicast/unicast mode they construct the GroupID from domain and sdoId and request a group key from the 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 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.

2.2.5.6. 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], chapter 8.5 "Initial Verification of Server certificates" which applies to client 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 KE server before requesters can request a unicast key (and ticket).

2.3. Overview of NTS Messages and their Structure for Use with PTP

Section 2.1 described the principle communication sequences for PTP Key Request, PTP Registration Request and corresponding response messages. All messages follow the "NTS Key Establishment Process" stated in the first part (until the description of Fig. 3 starts) of chapter 4 of IETF RFC 8915 [RFC8915]:

> _"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 6_. 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 0-7 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]‘."

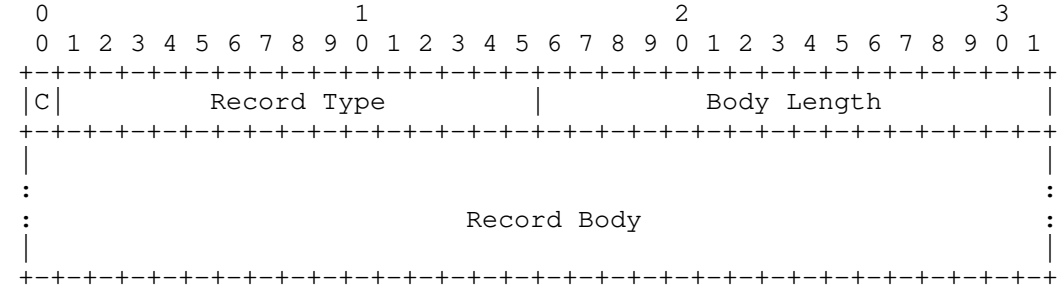


Figure 6: NTS-KE Record format

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 6. More details on record structure as well as the specific records used here are given in Section 3 and respective subsections there. 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 Parameter container.

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 has to be the last one in the sequence indicating the end of the current message. Container records do not include an End of Message record.

The NTS key management for PTP is based on six new NTS messages:

- PTP Key Request message (see Section 2.3.1)
- PTP Key Grant message (see Section 2.3.2)
- PTP Refusal message (see Section 2.3.3)
- PTP Registration Request message (see Section 2.3.4)
- PTP Registration Grant message (see Section 2.3.5)
- PTP Registration Revoke message (see Section 2.3.6)

The following sections describe the principle structure of those new NTS messages for the PTP key management. More details especially on the records the messages are built of and their types, sizes, requirements and restrictions are given in Section 3.2.

2.3.1. PTP Key Request Message

PTP Key Request

Record	Exemplary body contents
NTS Next Protocol Negotiation	PTPv2.1
NTS Message Version	1.0
NTS Message Type	PTP Key Grant
Current Parameters	set of Records {...}
MAC Algorithm Negotiation (optional)	{CMAC || HMAC}
Requesting PTP Identity (Unicast only)	data set {...}
End of Message	

Figure 7: Structure of a PTP Key Request message

Figure 7 shows the record structure of a PTP Key Request message. In the right column typical values are shown as examples. Detailed information on types, sizes etc. is given in Section 3.2. The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. Currently, the following NTS Message Version record always contains 1.0. The next record characterizes the message type, in this case PTP Key Request. The Association Mode record describes the mode how the PTP instance wants to communicate: In the group-based approach 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.

If there is an option to choose from additional MAC algorithms, then an optional record follows presenting the supported algorithms from which the KE server may choose. In ticket-based unicast mode, the Requesting PTP Identity 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.

2.3.2. PTP Key Grant Message

Figure 8 shows the record structure of a PTP Key Grant message. In the right column typical values are shown as examples. Detailed information on types, sizes etc. is given in Section 3.2. The

message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. Currently, the following NTS Message Version record always contains 1.0. The next record characterizes the message type, in this case PTP Key Grant.

PTP Key Grant

Record	Exemplary body contents
NTS Next Protocol Negotiation	PTPv2.1
NTS Message Version	1.0
NTS Message Type	PTP Key Grant
Current Parameters	set of Records {...}
Next Parameters	set of Records {...}
End of Message	

Figure 8: Structure of a PTP Key Grant 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. Figure 9 gives an example of data contained in that record. For more details on the records contained in the Current Parameters container see Section 3.2.3.

Current Parameters Container record (PTP Key Grant)

Record	Exemplary body contents
Security Policies	{(PTPmsg1||SPP:1)||(PTPmsg2||SPP:)}
Security Association	data set for SPP:1 {...}
[Security Association]	data set for SPP:2 {...}
Lifetime	1560s (=0h 26min)
Time until pdate	0s
Grace Period (optional)	10 seconds
Ticket Key ID (Unicast only)	156
Ticket (Unicast only)	data set {...}

Figure 9: Exemplary contents of a Current Parameters Container record of a PTP Key Grant message

If the request lies inside the update interval (i.e. TuU = 0, compare Figure 9), a Next Parameters Container record is appended giving all the security data needed in the upcoming validity period. Its structure follows the same composition as the Current Parameters record (in the ticked-based approach also including the Ticket Key ID record and the Ticket record). The messages always end with an End of Message record.

2.3.3. PTP Refusal Message

The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. Currently, the following NTS Message Version record always contains 1.0. The next record characterizes the message type, in this case PTP Refusal, see Figure 10. The Error record contains information about the reason of refusal. The messages always end with an End of Message record.

PTP Refusal

Record	Exemplary body contents
NTS Next Protocol Negotiation	PTPv2.1
NTS Message Version	1.0
NTS Message Type	PTP Refusal
Error	Association Port not registered
End of Message	

Figure 10: Structure of a PTP Refusal message

2.3.4. PTP Registration Request Message

PTP Registration Request

Record	Exemplary body contents
NTS Next Protocol Negotiation	PTPv2.1
NTS Message Version	1.0
NTS Message Type	PTP Registration Request
Requesting PTP Identity	data set {...}
AEAD Algorithm Negotiation	{AEAD_512 || AEAD_256}
MAC Algorithm Negotiation (optional)	{CMAC || HMAC}
End of Message	

Figure 11: Structure of a PTP Registration Request message

The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. Currently, the following NTS Message Version record always contains 1.0. The next record characterizes the message type, in this case PTP Registration Request, see Figure 11.

The Requesting PTP Identity record gives the addresses of the grantor requesting registration whereas the following AEAD Algorithm Negotiation record indicates which algorithms for encryption of the ticket the requester supports.

If there is an option to choose from additional MAC algorithms, then an optional record follows presenting all the grantor's supported algorithms from which the KE server may choose. The messages always end with an End of Message record.

2.3.5. PTP Registration Success Message

PTP Registration Success

Record	Exemplary body contents
NTS Next Protocol Negotiation	PTPv2.1
NTS Message Version	1.0
NTS Message Type	PTP Registration Success
Current Parameters	set of Records {...}
Next Parameters	set of Records {...}
End of Message	

Figure 12: Structure of a PTP Registration Success message

The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. Currently, the following NTS Message Version record always contains 1.0. The next record characterizes the message type, in this case PTP Registration Success, see Figure 12.

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. Figure 13 gives an example of data contained in that container as a response to PTP Registration Request. For more details on the records contained in the Current Parameters container see Section 3.2.3.

Current Parameters Container record (PTP Registration Success)

Record	Exemplary body contents
AEAD Algorithm Negotiation	AEAD_CMAC_512
Lifetime	2,460s (=0h 41min)
Time until pdate	0s
Ticket Key	{binary data}
Ticket Key ID	278
Grace Period (optional)	10 seconds

Figure 13: Exemplary contents of a Current Parameters Container record of a PTP Registration Success message

If the registration request lies inside the update interval a Next Parameters Container record is appended giving all the security data needed in the upcoming validity period. Its structure follows the same composition as the Current Parameters record. The messages always end with an End of Message record.

2.3.6. PTP Registration Revoke Message

PTP Registration Revoke

Record	Exemplary body contents
NTS Next Protocol Negotiation	PTPv2.1
NTS Message Version	1.0
NTS Message Type	PTP Registration Revoke
End of Message	

Figure 14: Structure of a PTP Registration Revoke message

The message starts with the NTS Next Protocol Negotiation record which in this application always holds PTPv2.1. Currently, the following NTS Message Version record always contains 1.0. The next

record characterizes the message type, in this case PTP Registration Revoke, see Figure 14. The messages always end with an End of Message record.

3. NTS Messages for PTP

This chapter covers the structure of the NTS messages and the details of the respective payload. The individual parameters are transmitted by NTS records, which are described in more detail in Section 3.2. In addition to the NTS records defined for NTP in IETF RFC8915, further records are required, which are listed in Table 1 below and begin with Record Type 1024 (compare IETF RFC 8915 [RFC8915], 7.6. Network Time Security Key Establishment Record Types Registry).

NTS Record Types	Description	Reference
0	End of Message	[RFC8915], section 4.1.1, this document, Section 3.2.4
1	NTS Next Protocol Negotiation	[RFC8915], section 4.1.2, this document, Section 3.2.12
2	Error	[RFC8915], section 4.1.3, this document, Section 3.2.5
3	Warning	[RFC8915], section 4.1.4
4	AEAD Algorithm Negotiation	[RFC8915], section 4.1.5, this document, Section 3.2.1
5	New Cookie for NTPv4 (not needed for PTP)	[RFC8915], section 4.1.6
6	NTPv4 Server Negotiation (not needed for PTP)	[RFC8915], section 4.1.7
7	NTPv4 Port Negotiation (not needed for PTP)	[RFC8915], section 4.1.8
8 - 1023	Reserved for NTP	
1024	Association Mode	This document, Section 3.2.2
1025	Current Parameters Container	This document, Section 3.2.3
1026	Grace Period	This document, Section 3.2.6
1027	Lifetime	This document,

		Section 3.2.7
1028	MAC Algorithm Negotiation	This document, Section 3.2.8
1029	Next Parameters Container	This document, Section 3.2.9
1030	NTS Message Type	This document, Section 3.2.10
1031	NTS Message Version	This document, Section 3.2.11
1032	Requesting PTP Identity	This document, Section 3.2.13
1033	Security Association	This document, Section 3.2.14
1034	Security Policies	This document, Section 3.2.15
1035	Ticket	This document, Section 3.2.16
1036	Ticket Container	This document, Section 3.2.17
1037	Ticket Key	This document, Section 3.2.18
1038	Ticket Key ID	This document, Section 3.2.19
1039	Time until Update	This document, Section 3.2.20
1040 - 16383	Unassigned	
16384 - 32767	Reserved for Private or Experimental Use	[RFC8915]

Table 1: NTS Key Establishment record types registry

3.1. NTS Message Types

This section repeats the composition of the specific NTS messages for the PTP key management in overview form. The specification of the respective records from which the messages are constructed follows in Section 3.2. The reference column in the tables refer to the specific subsections.

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 is mandatory the final record.

PTP Key Request

NTS Record Name	Comm. Type*	Use	Reference
NTS Next Protocol Negotiation	Multicast / Unicast	mand.	This document, Section 3.2.12
NTS Message Version	Multicast / Unicast	mand.	This document, Section 3.2.11
NTS Message Type	Multicast / Unicast	mand.	This document, Section 3.2.10
Association Mode	Multicast / Unicast	mand.	This document, Section 3.2.2
MAC Algorithm Negotiation	Unicast	opt.	This document, Section 3.2.8
Requesting PTP Identity	Unicast	mand.	This document, Section 3.2.13
End of Message	Multicast / Unicast	mand.	This document, Section 3.2.4

* The Communication Type column refers to the intended use of the particular record for the respective PTP communication mode.

Table 2: Record structure of the PTP Key Request message

PTP Key Grant

NTS Record Name	Comm. Type	Use	Reference
NTS Next Protocol Negotiation	Multicast / Unicast	mand.	This document, Section 3.2.12
NTS Message Version	Multicast / Unicast	mand.	This document, Section 3.2.11
NTS Message Type	Multicast / Unicast	mand.	This document, Section 3.2.10
Current Parameters Container	Multicast / Unicast	mand.	This document, Section 3.2.3
Next Parameters Container	Multicast / Unicast	opt. (conditional)	This document, Section 3.2.9
End of Message	Multicast / Unicast	mand.	This document, Section 3.2.4

Table 3: Record structure of the PTP Key Grant message

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

NTS Record Name	Comm. Type	Use	Reference
Security Policies	Multicast / Unicast	mand.	This document, Section 3.2.15
Security Association (one or more)	Multicast / Unicast	mand.	This document, Section 3.2.14
Lifetime	Multicast / Unicast	mand.	This document, Section 3.2.7
Time until Update	Multicast / Unicast	mand.	This document, Section 3.2.20
Grace Period	Multicast / Unicast	opt.	This document, Section 3.2.6
Ticket Key ID	Unicast	mand.	This document, Section 3.2.19
Ticket	Unicast	mand.	This document, Section 3.2.16

Table 4: Record structure of the container records

The encrypted Ticket Container within the Ticket record also includes a set of records listed below:

NTS Record Name	Comm. Type	Use	Reference
Requesting PTP Identity	Unicast	mand.	This document, Section 3.2.13
Security Policies	Multicast / Unicast	mand.	This document, Section 3.2.15
Security Association (one or more)	Multicast / Unicast	mand.	This document, Section 3.2.14
Lifetime	Multicast / Unicast	mand.	This document, Section 3.2.7
Time until Update	Multicast / Unicast	mand.	This document, Section 3.2.20
Grace Period	Multicast / Unicast	opt.	This document, Section 3.2.6

Table 5: Record structure of the enctypted Ticket container record

PTP Refusal

NTS Record Name	Comm. Type	Use	Reference
NTS Next Protocol Negotiation	Multicast / Unicast	mand.	This document, Section 3.2.12
NTS Message Version	Multicast / Unicast	mand.	This document, Section 3.2.11
NTS Message Type	Multicast / Unicast	mand.	This document, Section 3.2.10
Error	Multicast / Unicast	mand.	This document, Section 3.2.5
End of Message	Multicast / Unicast	mand.	This document, Section 3.2.4

Table 6: Record structure of the PTP Refusal message

PTP Registration Request

NTS Record Name	Comm. Type	Use	Reference
NTS Next Protocol Negotiation	Multicast / Unicast	mand.	This document, Section 3.2.12
NTS Message Version	Multicast / Unicast	mand.	This document, Section 3.2.11
NTS Message Type	Multicast / Unicast	mand.	This document, Section 3.2.10
Requesting PTP Identity	Unicast	mand.	This document, Section 3.2.13
AEAD Algorithm Negotiation	Unicast	mand.	This document, Section 3.2.1
MAC Algorithm Negotiation	Unicast	opt.	This document, Section 3.2.8
End of Message	Multicast / Unicast	mand.	This document, Section 3.2.4

Table 7: Record structure of the PTP Registration Request message

PTP Registration Success

NTS Record Name	Comm. Type	Use	Reference
NTS Next Protocol Negotiation	Multicast / Unicast	mand.	This document, Section 3.2.12
NTS Message Version	Multicast / Unicast	mand.	This document, Section 3.2.11
NTS Message Type	Multicast / Unicast	mand.	This document, Section 3.2.10
Current Parameters Container	Multicast / Unicast	mand.	This document, Section 3.2.3
Next Parameters Container	Multicast / Unicast	mand. (conditional)	This document, Section 3.2.9
End of Message	Multicast / Unicast	mand.	This document, Section 3.2.4

Table 8: Record structure of the PTP Registration Success message

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

NTS Record Name	Comm. Type	Use	Reference
AEAD Algorithm Negotiation	Unicast	mand.	This document, Section 3.2.1
Lifetime	Multicast / Unicast	mand.	This document, Section 3.2.7
Time until Update	Multicast / Unicast	mand.	This document, Section 3.2.20
Grace Period	Multicast / Unicast	opt.	This document, Section 3.2.6
Ticket Key ID	Unicast	mand.	This document, Section 3.2.19
Ticket	Unicast	mand.	This document, Section 3.2.16

Table 9: Record structure of the container records in th PTP Regsitration Success message

PTP Registration Revoke

NTS Record Name	Comm. Type	Use	Reference
NTS Next Protocol Negotiation	Multicast / Unicast	mand.	This document, Section 3.2.12
NTS Message Version	Multicast / Unicast	mand.	This document, Section 3.2.11
NTS Message Type	Multicast / Unicast	mand.	This document, Section 3.2.10
End of Message	Multicast / Unicast	mand.	This document, Section 3.2.4

Table 10: Record structure of the PTP Registration Revoke message

3.2. NTS Records

The following subsections describe the specific NTS records used to construct the NTS messages for the PTP key management system in detail. They appear in alphabetic sequence of their individual names. See Section 3.1 for the application of the records in the respective messages.

Note: For easier editing of the content, most of the descriptions in the following subsections are written as bullet points.

Global rules:

- The NTS Next Protocol Negotiation record MUST offer (at least) Protocol ID 1 for "PTPv2.1" (see Section 3.2.12).
- The NTS Message Version record MUST be v1.0.
- Note: Records must be used only in the mentioned messages. Not elsewhere.
- The notational conventions of Section 1 MUST be followed.

3.2.1. AEAD Algorithm Negotiation

This record is required in unicast mode and enables the negotiation of the AEAD algorithm needed to encrypt and decrypt the ticket. The negotiation takes place between the PTP grantor and the NTS-KE server by using the NTS registration messages. The structure and properties follow the record defined in IETF RFC 8915 [RFC8915], 4.1.5.

Content and conditions:

- The record has a Record Type number of 4 and the Critical Bit MAY be set.
- The record body contains a sequence of 16-bit unsigned integers in network byte order:

  Supported AEAD Algorithms = {AEAD 1 || AEAD 2 || ...}

- Each integer represents a numeric identifier of an AEAD algorithm registered by the IANA. (https://www.iana.org/assignments/aead-parameters/aead-parameters.xhtml)
- Duplicate identifiers SHOULD NOT be included.
- Grantor and NTS-KE server MUST support at least the AEAD_AES_SIV_CMAC_256 algorithm.
- A list of recommended AEAD algorithms is shown in the following table.
- Other AEAD algorithms MAY also be used.

Numeric ID	AEAD Algorithm	Use	Key Length (Octets)	Reference
15	AEAD_AES_SIV_CMAC_256	Mand.	16	[RFC5297]
16	AEAD_AES_SIV_CMAC_385	Opt.	24	[RFC5297]
18	AEAD_AES_SIV_CMAC_512	Opt.	32	[RFC5297]
32 - 32767	Unassigned			
32768 - 65535	Reserved for Private or Experimental Use			[RFC5116]

Table 11: AEAD algorithms

- In a PTP Registration Request message, this record MUST be contained exactly once.
- In this 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 Success 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 MUST be present only during the update period.

- The KE server SHOULD choose the highest priority AEAD algorithm from the request message that grantor and KE server support.
- The KE server MAY ignore the priority and choose a different algorithm that grantor and KE server support.
- In a PTP Registration Success message, this record MUST contain exactly one AEAD algorithm.
- The selected algorithm MAY differ in the Current Parameters Container and Next Parameters Container records.

3.2.2. Association Mode

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 two data fields:

Field	Octets	Offset
Association Type	2	0
Association Value	A	2

Table 12: Association

- 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 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

Unicast*: predefined groups of two (Group-of-2, Go2, see Group entry below)

Table 13: Association Types

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, 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 2.2.5.4).
- If no subgroups are required (=multicast mode), 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 equvalent 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)	subGroup (low)	2	4

Table 14: 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 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, 5.3.5, 7.5

3.2.3. Current Parameters Container

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 Grant 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	Comunication Type	Use	Reference
Security Policies	Multicast / Unicast	Mand.	This document, Section 3.2.15
Security Associations (one or more)	Multicast / Unicast	Mand.	This document, Section 3.2.14
Lifetime	Multicast / Unicast	Mand.	This document, Section 3.2.7
Time until Update	Multicast / Unicast	Mand.	This document, Section 3.2.20
Grace Period	Multicast / Unicast	Opt.	This document, Section 3.2.6
Ticket Key ID	Unicast	Mand.	This document, Section 3.2.19
Ticket	Unicast	Mand.	This document, Section 3.2.16

Table 15: Current Parameters Container for PTP Key Grant message

- The records Security Policies, Lifetime and Time until Update MUST be contained exactly once.
- The number of the Security Association records depends on the content of the Security Policies record (see Section 3.2.15).
- At least one Security Association record MUST be included.
- The Grace Period record is optional and MAY be absent.
- If it is present, it MUST be included exactly once.
- In order to establish a unicast connection with the PTP Key Grant message, the records Ticket Key ID and Ticket MUST be contained exactly once.
- If the requester wants to join a multicast group, the records Ticket Key ID and Ticket MUST NOT be included.

- In a PTP Registration Success message, the Current Parameters Container record MUST be contained exactly once.
- The record body MAY contain the following records:

NTS Record Name	Use	Reference
AEAD Algorithm Negotiation	Mand.	This document, Section 3.2.1
Lifetime	Mand.	This document, Section 3.2.7
Time until Update	Mand.	This document, Section 3.2.20
Grace Period	Opt.	This document, Section 3.2.6
Ticket Key ID	Mand.	This document, Section 3.2.19
Ticket	Mand.	This document, Section 3.2.16

Table 16: Current Parameters Container for PTP Registration Success Message

- The records AEAD Algorithm Negotiation, Lifetime, Time until Update, Ticket Key ID and Ticket Key MUST be contained exactly once.
- The Grace Period record is optional and MAY be absent.
- If it is present, it MUST be included exactly once.

3.2.4. End of Message

The End of Message record is defined in IETF RFC8915 [RFC8915], 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 request and response message.
- This record SHOULD NOT be included in the container records and MUST be ignored if present.
- See also: IETF RFC8915, 4.1.1

3.2.5. Error

The Error record is defined in IETF RFC8915 [RFC8915], 4.1.3. In addition to the Error codes 0 to 2 specified there the following Error codes 3 to 4 are defined:

Error Code	Description
0	Unrecognized Critical Record
1	Bad Request
2	Internal Server Error
3	Requester not Authorized
4	Grantor not Registered
5 - 32767	Unassigned
32768 - 65535	Reserved for Private or Experimental Use

Table 17: 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 "Requester not Authorized" is sent by the KE server if the requester is not authorized to join the desired multicast group.
- This Error code MUST NOT be included as a response to PTP Registration Request message.

- The Error code 4 "Grantor not Registered" is sent by the KE server when the requester wants to establish a unicast connection to a grantor that is not registered with the KE server.
- This Error code MUST NOT be included as a response to a PTP Key Request message.

3.2.6. Grace Period

The Grace Period determines the time period in which expired security parameters may still be accepted. It allows the verification of PTP messages, which have been secured with the previous key at the rotation time of the security parameters.

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.
- This value contains the transition time in seconds in which an expired key MAY still be accepted.

- o A time of zero seconds is valid.
- o If this optional record is absent, a default time of zero seconds is used unless a PTP profile defines something else.

- o The Grace Period record MAY only appear as part of a PTP Key Grant or PTP Registration Success message.
- o In a PTP Key Grant message, the Grace Period MAY be in the Current Parameters Container and Next Parameters Container records, as well as a part of the encrypted Ticket Container (if present).
- o The Grace Period record MUST NOT appear more than once in each container or ticket.

- o In a PTP Registration Success message, the Grace Period record MAY be present in the Current Parameters Container record as well as in the Next Parameters Container record.
- o The Grace Period MUST NOT be included more than once in each of those container records.

- o The Next Parameters Container MUST be present only during the update period.

3.2.7. Lifetime

This record specifies the lifetime of a defined set of parameters. The value contained in this record is counted down by the receiver of the NTS message every second. When the value reaches zero, the parameters associated with this record are considered to have expired.

Content and conditions:

- o The record has a Record Type number of 1027 and the Critical Bit MAY be set.
- o The record body consists of a 32-bit unsigned integer in network byte order, denoting the expiration time of specific parameters in seconds.

- o The maximum value is set by the NTS-KE administrator or the PTP profile.
- o In conjunction with a PTP unicast establishment, the Lifetime of the unicast key, the ticket key and registration lifetime of a grantor with the KE server MUST be identical.

- o The Lifetime record MAY only appear as part of a PTP Key Grant or PTP Registration Success message.
- o In both messages, the Next Parameters Container MUST be present only during the update period.

- In a PTP Key Grant message, the Lifetime record MUST be included exactly once in the Current Parameters Container and Next Parameters Container records, as well as in the encrypted Ticket Container (only present in a unicast PTP Key Grant message).
- In a PTP Registration Success message, the Lifetime MUST be included exactly once in both records, Current Parameters Container and Next Parameters Container.

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.

3.2.8. MAC Algorithm Negotiation

This optional 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 only used in unicast mode.

Content and conditions:

- The record has a Record Type number of 1028 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 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 18: MAC Algorithms

In PTP multicast mode:

- 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 massage and the NTS-KE server MUST ignore this record.
- Unless this is specified by a PTP profile, the HMAC-SHA256-128 algorithm SHALL be used by default.

In PTP unicast mode:

- In a PTP Key Request message, this record MAY be contained if the requester wants a unicast connection 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.
- The default MAC algorithm (HMAC-SHA256-128) MAY be omitted in the record.

- In a PTP Registration Request message, this record MUST be present and the grantor MUST include all supported MAC algorithms in any order.

- The KE server selects the algorithm after receiving a PTP Key Request message in unicast mode.
- The KE server SHOULD choose the highest priority MAC algorithm from the request message that grantor and requester support.
- The 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 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, 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.

3.2.9. Next Parameters Container

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 1029 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 3.2.3 of this document.

- In both the PTP Key Grant and PTP Registration Success message, this record MUST be contained exactly once during the update period.
- Outside the update period, this record MUST NOT be included.
- In multicast mode, this record MAY also be missing if the requester is to be explicitly excluded from a multicast group after the security parameter rotation process by the KE server.

- The update period starts with the expiration of the Time until Update timer, which is stored in the Current Parameter Container record.
- In the PTP Key Grant and PTP Registration Success message, the expiration of the Lifetime marks the end of the update period.
- More details are described in Section 2.2.1.

3.2.10. NTS Message Type

This record enables the distinction between different NTS message types for PTP.

Content and conditions:

- The record has a Record Type number of 1030 and the Critical Bit MUST be set.
- The record body is a 16-bit unsigned integer in network byte order, denoting the type of the current NTS message for PTP.
- The message types are defined in the following table.
- More details about the messages are described in Section 2.3

NTS Message Type Number	NTS Message Name
0	PTP Key Request
1	PTP Key Grant
2	PTP Refusal
3	PTP Registration Request
4	PTP Registration Success
5	PTP Registration Revoke
6 - 32767	Unassigned
32768 - 65535	Reserved for Private or Experimental Use

Table 19: NTS message type numbers

3.2.11. NTS Message Version

This record enables the distinction between different NTS message versions for PTP. It provides the possibility to update or extend the NTS messages in future specifications.

Content and conditions:

- The record has a Record Type number of 1031 and the Critical Bit MUST be set.

- The record body consists of a tuple of two 8-bit unsigned integers in network byte order.
- The first octet represents the major version and the second octet the minor version.

  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 record body MUST match 0100h.

3.2.12. NTS Next Protocol Negotiation

The Next Protocol Negotiation record is defined in IETF RFC8915 [RFC8915], 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 as shown in the table below.
- For NTS requests messages for PTPv2.1, 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], 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, NTS4NTP 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.

3.2.13. Requesting PTP Identity

This record allows the KE server to associate an NTS unicast request of a requester with a registered grantor based on their address or identifier (e.g.: IP address or PortIdentity). Furthermore, this record allows the grantor to verify the origin of a secured PTP message that is currently transmitting a ticket.

Content and conditions:

- The record has a Record Type number of 1032 and the Critical Bit MAY be set.
- The record body consists of a set of Association Types together with their respective Association Values.

Field	Octets	Offset
Association Type 1	2	0
Association Value 1	A1	2
Association Type 2	2	A1+2
Association Value 2	A2	A1+4
Association Type n	A2	A1+A2+4
Association Value n	An	A1+A2+6

Table 21: Requesting PTP identity list

- Structure and values are based on the contents defined in Section 3.2.2 of this document.
  - Therefore, the Association Type is a 16-bit unsigned integer.
  - The length and content of Association Value depends on the value of Association Type.
  - All bytes are stored in network byte order and the rules in Section 3.2.2 MUST be followed.
- A Requesting PTP Identity record MUST contain at least one association tuple (type + value).
- This record can contain several association tuples in any order.
- It MUST NOT contain more than one association tuple of the same type.

- In a PTP Key Request message, this record MUST be contained exactly once in the unicast mode, which depends on the content of the Association Mode record of this message.
- In this case the Requesting PTP Identity record MUST contain exactly one association tuple.
- This association tuple MUST contain one identification feature of the PTP requestor (IPv4, IPv6, 802.3 or PortIdentity).
- The association tuple MUST NOT contain the Group association type 0.

- In a PTP Key Grant message, this record MUST be contained exactly once in the encrypted Ticket Container.
- This record MUST contain exactly one association tuple.
- The record body MUST be identical to the Requesting PTP Identity record of the related PTP Key Request message.
- Therefore, the association tuple MUST NOT contain the Group association type 0.

- In a PTP Registration Request message, this record MUST be included exactly once.

- The grantor SHOULD add the following association tuples as far as they are available: IPv4, IPv6, 802.3 and PortIdentity.
- The grantor MUST NOT include the Group association type 0.

- This allows a requester to be assigned to a grantor, regardless of whether the requester specifies IPv4, IPv6, 802.3 or the PortIdentity of the grantor in its PTP Key Request message.

3.2.14. Security Association

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, 16.14.3). Static values and flags, such as the secParamIndicator, are described in more detail in Section 5.

Content and conditions:

- The record has a Record Type number of 1033 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 22: Security Association record

- In a PTP Key Grant message, the Security Association record MUST be included at least once in the Current Parameters Container record and the Next Parameters Container record.
- In unicast mode, the Security Association record MUST be included at least once in the encrypted Ticket Container as well.
- The Next Parameters Container record MUST be present only during the update period.
- The Ticket record MUST be present in unicast mode and MUST NOT be present in multicast mode.
- The number of Security Association records in the respective container or Ticket Container depends on the content of the associated Security Policies (see also Section 3.2.15).

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 KE server (see Section 3.3.2).

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 3.2.8.
- 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 KE server.
- The NTS-KE server MUST ensure that every Key ID is unique.
    * The value can be either a random number or an enumeration.
    * Previous Key IDs SHOULD NOT be reused for a certain number of rotation periods or a defined period of time (see Section 3.3).

Key Length

- This value is a 16-bit unsigned integer in network byte order, denoting the length of the key.

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 (multicast mode) or a unicast key (unicast mode).

3.2.15. Security Policies

This record contains the information "which" PTP message types must be secured.

Content and conditions:

- The record has a Record Type number of 1034 and the Critical Bit MAY be set.
- The record body contains a sequence of tuples in network byte order:

  Record body = {Security Policies = {tuple 1 || tuple 2 || tuple 3 || tuple n}}

- Each tuple has a length of 2 octets and consists of a sequence of a PTP Message Type and a Security Parameter Pointer.

Field	Octets	Offset
PTP Message Type	1	0
Security Parameter pointers	1	1

Table 23: Security Policy tuple

- The PTP Message Type is an 8-bit unsigned integer.
- The most significant 4 bits are zero-padded and the least significant 4 bits are the PTP message type:

Structure of PTP Message Type (see also [IEEE1588-2019], 13.3.2.3, table 36):

Bits 7 - 4	Bits 3 - 0
Zero Padding	PTP Message type

Table 24: PTP Message Type

- The Security Parameter Pointer (SPP) is an 8-bit unsigned integer in the closed range 0 to 255.

- The record body MUST contain at least one tuple.
- A tuple associates a PTP message type with an SPP.
- Every PTP message type that is mentioned in the Security Policies record MUST be secured.
- Thus, a PTP message type that is not included in this record MUST NOT contain an AUTHENTICATION TLV and will not be secured.
- Multiple tuples with the same PTP message type MUST NOT be included.

- Multiple tuples MAY use the same SPP to use a shared security association or an individual one.

- For the number of contained and different SPPs in the Security Policies record, the same number of security associations MUST be created.
- The number of security associations determines the number of Security Associations records in the respective container record (e.g. Current Parameters Container).

- In a PTP Key Grant message, this record MUST be included exactly once each in the Current Parameters Container record, the Next Parameters Container record as well as the encrypted Ticket Container record.
- The Next Parameters Container record MUST be present only during the update period.
- The Ticket record MUST be present in unicast mode and MUST NOT be present in multicast mode.

3.2.16. Ticket

This record contains the parameters of the selected AEAD algorithm, as well as an encrypted Ticket Container record. The encrypted record contains all the necessary security parameters that the grantor needs for a secured PTP unicast connection to the requester. The ticket container 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 ticket container.

Content and conditions:

- The record has a Record Type number of 1035 and the Critical Bit MAY be set.
- The record body consists of several data fields and MUST be formatted as follows.

Field	Octets	Offset
Nonce Length	2	0
Nonce	N	2
Encrypted Ticket Container Length	2	N+2
Encrypted Ticket Container	C	N+4

Table 25: Structure of a Ticket record

- In a PTP Key Grant message, this record MUST be included exactly once each in the Current Parameters Container record and the Next Parameters Container record if the requester wants a unicast communication to a specific grantor.
- The Next Parameters Container record MUST be present only during the update period.

Nonce Length

- This is a 16-bit unsigned integer in network byte order, denoting the length of the Nonce field.

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 3.3.1.

Encrypted Ticket Container Length

- This is a 16-bit unsigned integer in network byte order, denoting the length of the Encrypted Ticket Container field.

Encrypted Ticket Container

- This field contains the output of the AEAD operation ("Ciphertext") after the encryption process of the respective Ticket Container record.
- The plaintext of this field is described in Section 3.2.17.
- More details about the AEAD process and the required input data are described in Section 3.3.1.

3.2.17. Ticket Container

This record is a simple container that can carry an arbitrary number of NTS records. It contains all relevant security parameters that a grantor needs for a secured unicast connection. 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. The Ticket Container record serves as input parameter for the AEAD operation (see Section 3.2.1) and is transmitted encrypted within the Ticket record (see Section 3.2.16).

Content and conditions:

- The record has a Record Type number of 1036 and the Critical Bit MAY be set.
- The record body is defined as a set of records and MAY contain the following records.

NTS Record Name	Use	Reference
Requesting PTP Identity	mand.	This document, Section 3.2.13
Security Policies	mand.	This document, Section 3.2.15
Security Association (one or more)	mand.	This document, Section 3.2.14
Lifetime	mand.	This document, Section 3.2.7
Time until Update	mand.	This document, Section 3.2.20
Grace Period	opt. (conditional)	This document, Section 3.2.6

Table 26: Structure of a Ticket Container

- The records Requesting PTP Identity, Security Policies, Lifetime and Time until Update MUST be contained exactly once.
- The number of the Security Association records depends on the content of the Security Policies record (see Section 3.2.15).
- All records within this Ticket Container (except Requesting PTP Identity) MUST be identical to the records of the respective Current Parameter Container.
- All records within this Ticket Container (except Requesting PTP Identity) MUST be identical to the records of the respective Next Parameter Container.
- The presence of the Grace Period record also depends on the respective Current/Next Parameter container.
- If a Grace Period record is present in the Current/Next Parameter container, it MUST also be present in the respective Ticket Container.
- If it is not present, it MUST NOT be included in the Ticket Container.

3.2.18. Ticket Key

This record contains the ticket key, which together with an AEAD algorithm is used to encrypt and decrypt the ticket.

Content and conditions:

- The record has a Record Type number of 1037 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 requirement of the associated AEAD algorithm.

- In a PTP Registration Success 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.

3.2.19. Ticket Key ID

The Ticket Key ID record is a unique identifier that allows a grantor to identify the associated ticket key.

Content and conditions:

- The record has a Record Type number of 1038 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 either a random number, a hash value or an enumeration.
  - Previous IDs SHOULD NOT be reused for a certain number of rotation periods or a defined period of time.
- In a PTP Key Grant message, this record MUST be included exactly once each in the Current Parameters Container record and the Next Parameters Container record if a unicast connection is to be established.
- If the requester wishes to join a multicast group, the Ticket Key ID record MUST NOT be included in the container records.
- In a PTP Registration Success 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 unicast mode and MUST NOT be present in multicast mode.

3.2.20. Time until Update

The Time until Update (TuU) record specifies the point in time at which new security parameters are available. The value contained in this record is counted down by the receiver of the NTS message every second. When the value reaches zero, the update period begins and NTS response messages typically contain the Next Parameter Container record for a certain period of time (see also Section 2.2.1).

Content and conditions:

- The record has a Record Type number of 1039 and the Critical Bit MAY be set.
- The record body consists of a 32-bit unsigned integer in network byte order, denoting the begin of the update period in seconds.

- The value in the TuU MUST be less than the value in the associated Lifetime record (in the same container or ticket).
- If the value in the TuU is greater than zero in the Current Parameter Container, the corresponding message MUST NOT contain a Next Parameters Container.
- If the value in the TuU is zero in the Current Parameters Container, the corresponding NTS message MAY contain the Next Parameters Container record.

- The Time until Update record MAY only appear as part of a PTP Key Grant or PTP Registration Success message.
- In both messages, the Next Parameters Container MUST be present only during the update period.
- In a PTP Key Grant message, the Time until Update record MUST be included exactly once each in the Current Parameters Container and Next Parameters Container records, as well as in the encrypted Ticket Container (only present in a unicast PTP Key Grant message).
- In a PTP Registration Success message, the Time until Update MUST be included exactly once each in the Current Parameters Container and Next Parameters Container records.
- In both messages, the Next Parameters Container record MUST be present only during the update period.

Notes:

- Requests during the currently running lifetime will receive respectively adapted count values for Time until Update.
- During the update period the value for TuU in the Current Parameters Container will be zero.

3.3. Additional Mechanisms

This section provides information about the use of the negotiated AEAD algorithm as well as the generation of the security policy pointers.

3.3.1. 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 3.2.1) 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 3.2.1.
- 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 Ticket Container record, this parameter MUST remain empty.
- Nonce:
  - Corresponds to the value from the Nonce field in the Ticket (Section 3.2.16).
  - The requirements and conditions depend on the selected AEAD algorithm.
  - For the AEAD algorithms defined in Section 3.2.1 (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 Ticket Container 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 Ticket Container record with all records inside.
  - This is also the output of the AEAD operation after the decryption process.
- Ciphertext:

* Corresponds to the value from the Encrypted Ticket Container field in the Ticket (Section 3.2.16).
* The Ciphertext is the output of the AEAD operation after the encryption process.
* This is also the input parameter for the AEAD decryption operation.

3.3.2. SA/SP Management

This section describes the requirements and recommendations attached to SA/SP 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 B.
- 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 Parameter). 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 Pseudorandom Number Generator (CSPRNG) on the NTS-KE server (see also Section 2.2.2). The length of the keys depends on the MAC algorithm used. The generation and management of the Key ID is also controlled by the 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 Parameter).

4. 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 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 registered port. With the reception of the key grant the requester obtains the unicast key and the Ticket record containing the encrypted ticket container (see Section 2.1.2 and Section 3.2.16). The ticket container (see Section 3.2.17) includes the identification of the requester, the SAs along with the unicast key as well as the Lifetime/Time until Update data.

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 container 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 container 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 requester's identity does not match with the Requesting PTP Identity record in the Ticket Container and/or the ICV in the AUTHENTICATION TLV is not identical to the generated ICV by the grantor, then the unicast request message shall be denied.

The TICKET TLV structure is given in Table 27 below.

Field	Octets	Offset
tlvType	2	0
lengthField	2	2
Ticket record	T	4

Table 27: Structure of the TICKET TLV

To comply with the TLV structure of IEEE Std 1588-2019 ([IEEE1588-2019], 14.1) the TICKET TLV is structured as presented in Table 27 with a newly defined tlvType, a respective length field and the Ticket record (see Section 3.2.16) containing the encrypted Ticket container. Eventually it may be necessary to define the Ticket TLV externally to IEEE 1588 SA. Then the structure should follow IEEE Std 1588-2019 ([IEEE1588-2019], 14.3) to define a new standard organization extension TLV as presented in Table 28 below.

Field	Octets	Offset
tlvType	2	0
lengthField	2	2
organizationId	3	4
organizationSubType	3	7
Ticket record	T	10

Table 28: Structure of an organization extension TLV form for the TICKET TLV

To transport the TICKET TLV with the Ticket container embedded via the PTP unicast negotiation message two possible solutions exist:

a. The TICKET TLV can be added to the PTP message preceding the AUTHENTICATION TLV as shown in Figure 48 of IEEE Std 1588-2019 ([IEEE1588-2019], 16.14.1.1). For this solution, a completely new TICKET TLV for IEEE Std 1588-2019 needs to be defined.
b. In an alternative solution the TICKET TLV is send embedded in the RES field of the AUTHENTICATION TLV as shown in Figure 49 of IEEE

Std 1588-2019 ([IEEE1588-2019], 16.14.3). In this case the RP flag in the secParamIndicator must be set. As at the moment the use of the RES field is not permitted and the structure of the RES field is limited to UInteger (see [IEEE1588-2019], 16.14.3.8) the new usage needs to be defined:

"16.14.3.8 RES (UInteger R): This field is optional. If present, it shall have a data type of UInteger with a length of R octets. For this edition, the value of RP in the secParamIndicator field shall be FALSE and the value of RP shall be 0."

Which solution is chosen is a political question, not a technical one and needs to be discussed in the IEEE 1588 SA. The same applies to the format of the TICKET TLV (standard TLV or organization extension TLV).

5. 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 29 below (compare to Figure 49 of [IEEE1588-2019]).

Field	Use	Description
tlvType	mand.	TLV Type
lengthField	mand.	TLV Length Information
SPP	mand.	Security Parameter Pointer
secParamIndicator	mand.	Security Parameter Indicator
keyID	mand.	Key Identifier or Current Key Disclosure Interval, depending on verification scheme
disclosedKey	opt.	Disclosed key from previous interval
sequenceNo	opt.	Sequence number
RES	opt.	Reserved
ICV	mand.	ICV based on algorithm OID

Table 29: 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 KE server in the PTP Key Grant Message

The optional disclosedKey, sequenceNo, and RES (see discussion in chapter 3) fields are omitted. So all of the flags in the SecParamIndicator are FALSE.

ICV field contains the integrity check value of the particular PTP message calculated using the integrity algorithm defined by the key management.

6. IANA Considerations

Considerations should be made ...

...

7. Security Considerations

...

8. Acknowledgements

The authors would like to thank ...

9. References

9.1. 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.

[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>.

[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>.

[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>.

[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>.

[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)", BCP 195, 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>.

9.2. 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>.

Authors' Addresses

Martin Langer
Ostfalia University of Applied Sciences
Salzdahlumer Strasse 46/48
Wolfenbuettel 38302
Germany

Email: mart.langer@ostfalia.de

Rainer Bermbach
Ostfalia University of Applied Sciences
Salzdahlumer Strasse 46/48
Wolfenbuettel 38302
Germany

Email: r.bermbach@ostfalia.de

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2021

M. Lichvar
Red Hat

Feb 17, 2021

Network Time Protocol Version 5
draft-mlichvar-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 August 21, 2021.

Table of Contents

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 a new measurement 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.

11. Clients don’t 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 prevent 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 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 -1.0, the maximum value is 32767/32768, and the resolution is about 30 microseconds.

time32
: A 32-bit fixed-point type containing values in seconds. It has 4 unsigned integer bits and 28 fractional bits. The maximum value is 16 seconds and the resolution is about 3.7 nanoseconds.

timestamp64
: A 64-bit fixed-point type containing timestamps. 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, 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. 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.

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.

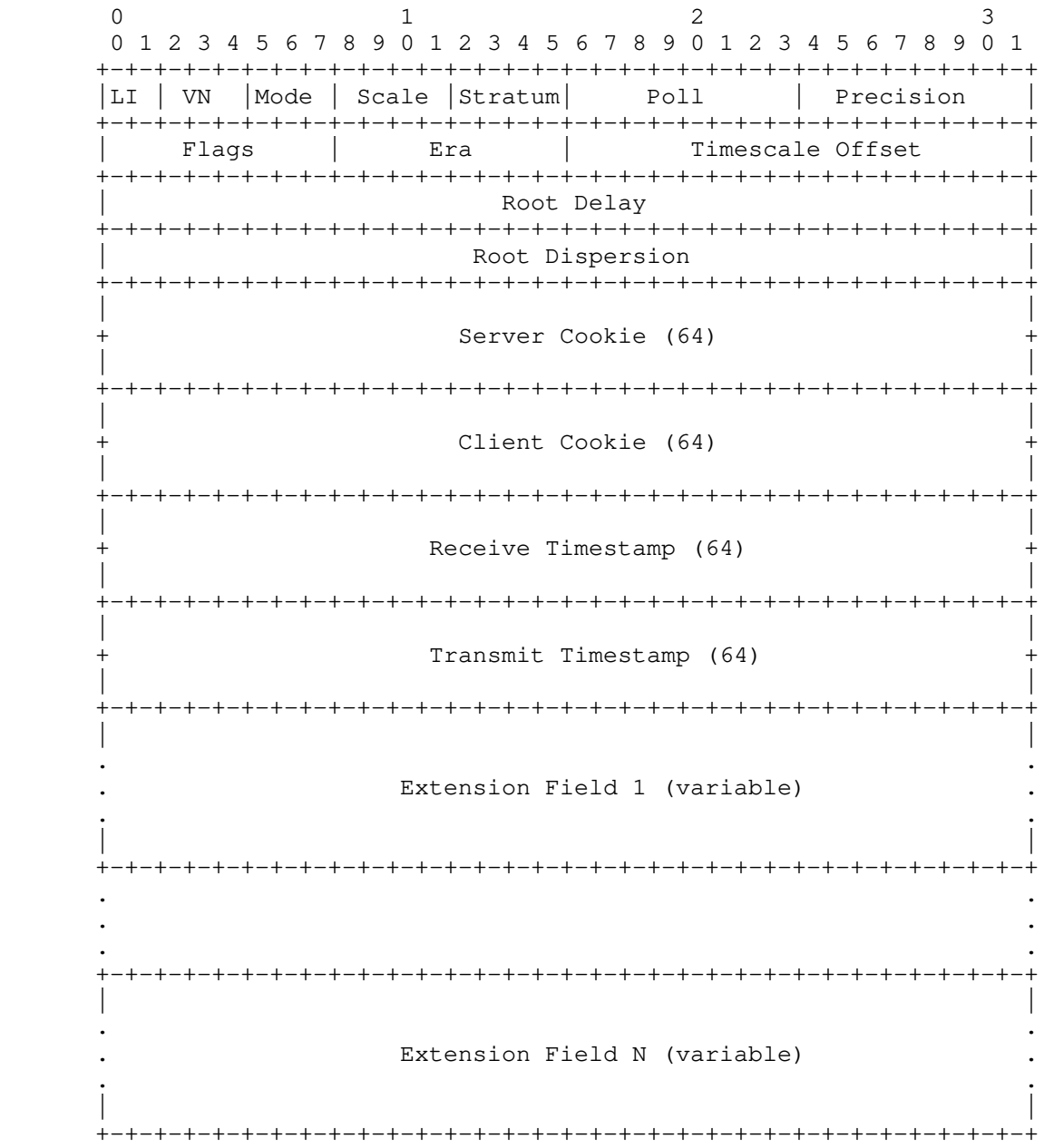


Figure 1: Format of NTPv5 messages

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. 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).

Scale
A 4-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-smeared UTC

Stratum
A 4-bit field containing the stratum of host. 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.

Poll
An 8-bit signed integer containing the polling interval as a rounded log2 value in seconds. In requests it is the current polling interval. In responses it is the minimum allowed polling interval.

Precision
An 8-bit signed integer containing the precision of the timestamps included in the message as a rounded log2 value in seconds. In requests, which don't contain any timestamps, it is always 0.

Flags
An 8-bit integer that can contain the following flags:

0x1: Unknown leap

In requests it is zero. In responses it 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 values: synchronized (0) and not synchronized (3).

0x4: Interleaved mode
In requests it is a request for a response in the interleaved mode. In responses it indicates the response is in the interleaved mode.

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

Timescale Offset
A 16-bit value specific to the selected timescale, which is referenced to the receive timestamp. In requests it is always 0.

* In the UTC (0) and TAI (1) timescales it is the TAI-UTC offset as a signed integer, or 0x8000 if unknown.

* In the UT1 timescale (2) it is the UT1-UTC offset using the time16 type, or 0x8000 (-1.0) if unknown.

* In the leap-smeared UTC, it is the current offset between the leap smeared time and UTC using the time16 type, or 0x8000 (-1.0) if unknown.

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 valid responses 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. 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 time when a response to the client was transmitted. The specific response depends on the selected mode (basic or interleaved). The timestamp corresponds to the beginning of the transmission.

The header has 48 octets, which is the minimum length of a valid NTPv5 message. A message can contain zero, one, or multiple extension fields. The maximum length is not specified, but the length is always divisible by 4.

5. Extension Fields

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

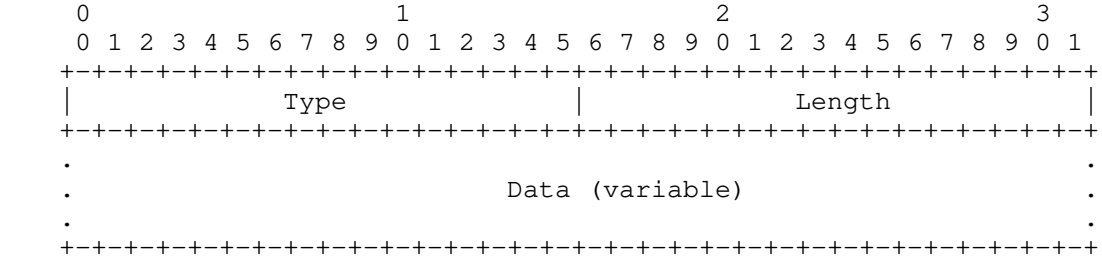


Figure 2: Format of NTPv5 extension fields

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 doesn't have to contain any data. If the length is not divisible by 4, the extension field is padded with zeroes to the smallest multiple of 4 octets.

Generally, if a request contains an extension field, the client is asking the server to include the same extension field in the response. Exceptions to this rule are allowed.

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

The rest of this section describes new extension fields specified for NTPv5. Clients are not required to use or support any of these

extension fields, but servers are required to support some extension fields.

5.1. Padding Extension Field

This field is used by servers to pad the response to the same length as the request if the response doesn't contain all requested extension fields, or some have a variable length. It can have any length.

This field MUST be supported on server.

5.2. MAC Extension Field

This field 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.

5.3. Reference IDs Extension Field

This field allows servers to prevent synchronization loops, i.e. synchronizing to one of its direct or indirect clients. It contains a set (bloom filter) of reference IDs.

TODO

This field MUST be supported on server.

5.4. Server Information Extension Field

This field provides clients with information about which NTP versions are supported by the server, as a minimum and maximum version. The extension field has a fixed length of 8 octets. In requests, all data fields of the extension are 0.

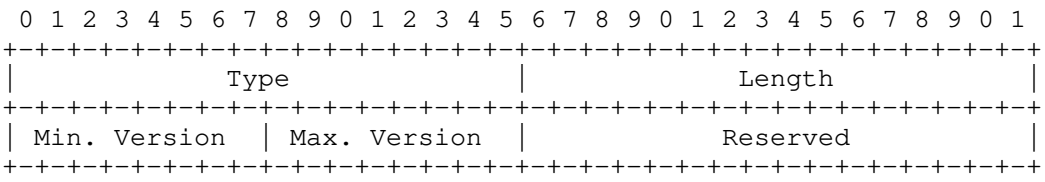


Figure 3: Format of Server Information Extension Field

This field MUST be supported on server.

5.5. 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 acccount for the accumated Delay Correction to determine a more accurate clock offset.

The NTPv5 Delay Correction has the same format as the PTP correctionField to make it easier for manfacturers of switches and routers to implement NTP corrections. The format of the Correction Extension Field is shown in Figure 4.

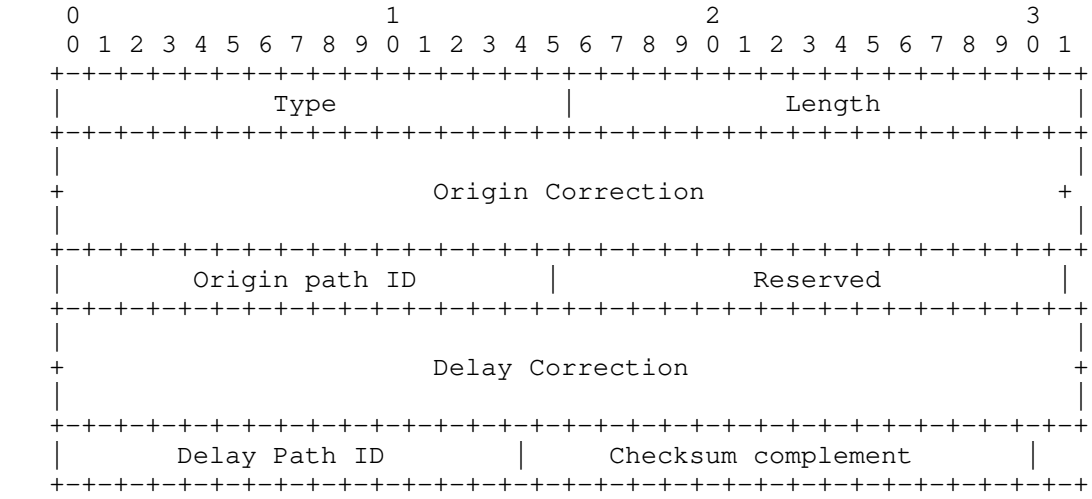


Figure 4: Format of Correction Extension Field

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 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].

A correction capable client SHALL transmit the request with the Origin Correction, Origin ID, Delay Correction and Path ID fields filled with all zeros.

Network nodes, such as switches and routers, that are NTP corrections capable SHALL add the difference between the beginning of an NTP message retransmission and the end of the message reception to the received Delay Correction value, and update this field. Note that this time difference might be negative, for example 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. Both timestamps MUST be based on the same clock. This clock does not need to be synchronized as long as the frequency is accurate enough such

that resulting time difference estimation errors are acceptable to the precision required by the application.

If a network node updates the delay correction, it SHOULD also add the identification numbers of the incoming and outgoing port to the path ID. Path ID values can be used by clients to determine if the ntp request and response messages are likely to have traversed the same network path.

If a network node modified any field of the extension field, it MUST update the checksum complement field in order to keep the current UDP checksum valid, or update the UDP checksum itself.

The server SHALL write the received Delay Correction value in the origin correction field of the response message, and the recieved path ID value in the origin ID field. The server SHALL set the Delay Correction field and Path ID fields to all zeros

5.6. Reference Timestamp Extension Field

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

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

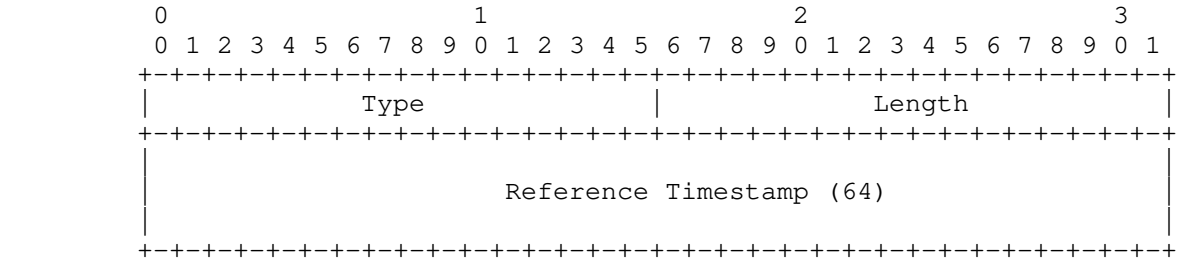


Figure 5: Format of Reference Timestamp Extension Field

5.7. Monotonic 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 transfering time and frequency separately using different clocks.

The Monotonic Timestamp Extension Field contains an extra receive timestamp with a 32-bit epoch identifier captured by a clock which doesn't 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 identifier is a random number which is changed when frequency transfer needs to be restarted, e.g. due to a step of the clock.

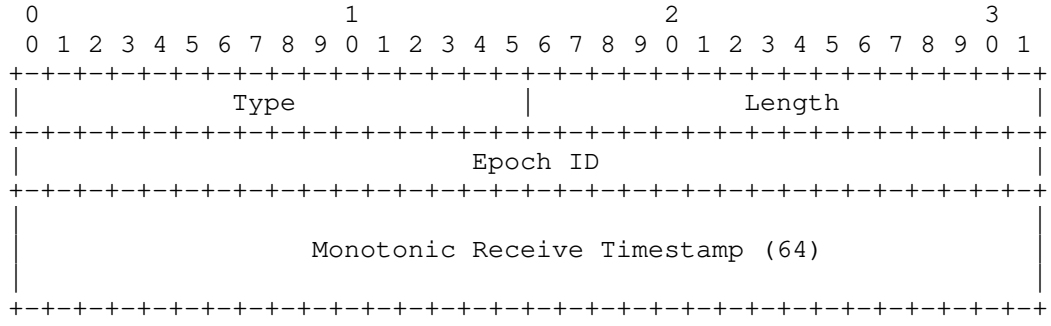


Figure 6: Format of Monotonic Timestamp Extension Field

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.

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) + (Cd - Co)) / 2
- delay = |(T4 - T1) - (T3 - T2) - (Cd + Co)|
- dispersion = |T4 - T1| * DR

where

- T1, T2, T3, T4 are the receive and transmit timestamps of a request and response
- Co is the Origin Correction from the Correction Extension Field if present in the response and has acceptable values, zero otherwise
- Cd is the Delay Correction from the Correction Extension Field if present in the response and has acceptable values, zero otherwise
- DR is the client's dispersion rate

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 indentified 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 previous response was formed or sent.

An example of cookies and timestamps in an NTPv5 exchange using the basic mode is shown in Figure 7.

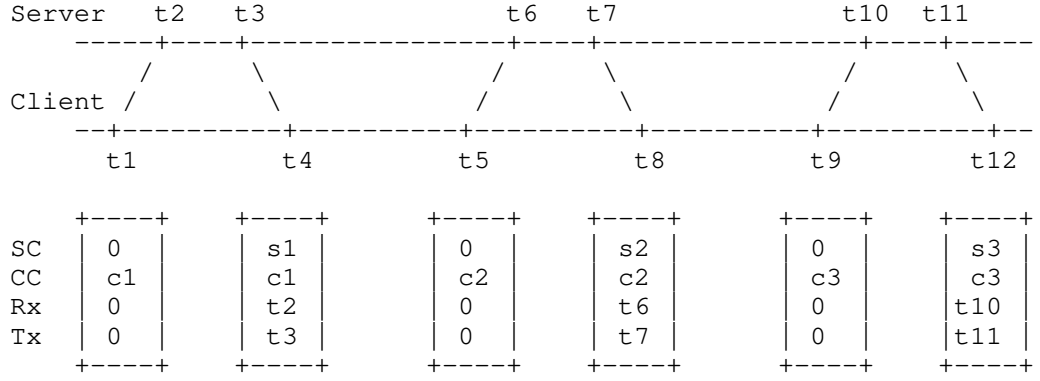


Figure 7: Cookies and timestamps in basic mode

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 a random cookie instead of the receive timestamp.

An example of an NTPv5 exchange using the interleaved mode is shown in Figure 8. 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. 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).

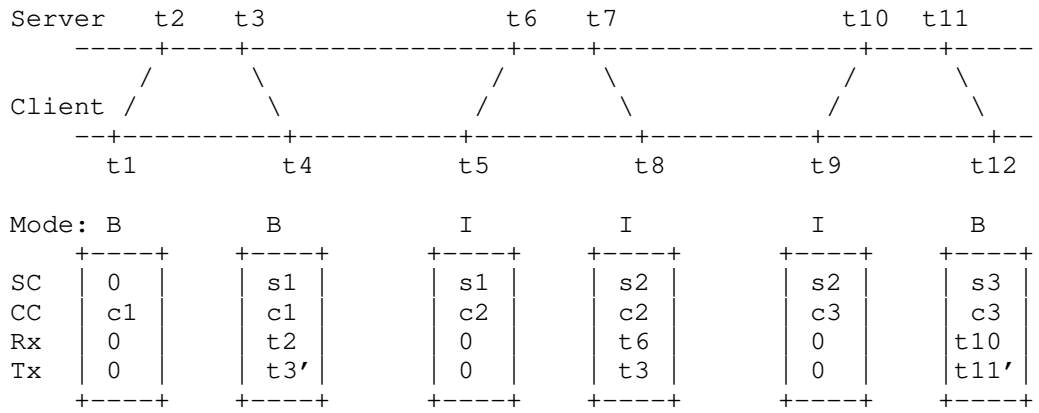


Figure 8: Cookies and timestamps in interleaved mode

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 at least a polling interval of 64 seconds, increasing up to at least 1024 seconds.

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 a response in the interleaved mode.
    * Server cookie is set only in the interleaved mode. If a valid response from the server was received previously, it is set to the server cookie from the previous response.

* 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.

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 provided or known, or drop 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.

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 doesn't 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 day 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 reference ID and it MUST track reference IDs of its servers using the Reference IDs 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. The server MAY respond with an NTPv5 message if and only if the request has version 5.

3. Drops the request if the format is not valid, mode is not 3, or authentication fails if the MAC Extension Field or another authenticator field is present. 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 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.

* Timescale Offset is set to the timescale-specific offset, or 0x8000 if unknown.

* 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 yet due to the request having an invalid server cookie. The cookie identifies a more accurate transmit timestamp, which can be retrieved by the client later with another request.

* 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 if 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. NTPv5 Negotiation in NTPv4

NTPv5 messages are not compatible with NTPv4, even if they do not contain any extension fields. Some widely used NTPv4 implementations are known to ignore the version and interpret all requests as NTPv4. Their responses to NTPv5 requests have a zero client cookie, which means they fail the client's validation and are ignored.

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 reference timestamp field in the NTPv4 header is reused for this purpose.

An NTP server which supports both NTPv4 and NTPv5 SHOULD check the reference timestamp in all NTPv4 client requests. If the reference

timestamp contains the value 0x4E5450354E545035 ("NTP5NTP5" in ASCII), it SHOULD respond with the same reference timestamp to indicate it supports NTPv5.

An NTP client which supports both NTPv4 and NTPv5, and is not configured to use a particular version, SHOULD start with NTPv4 requests having the reference timestamp set to 0x4e5450354e545035. If the server responds with the same reference timestamp, the client SHOULD switch to NTPv5.

10. Acknowledgements

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

The author would like to thank Doug Arnold for his contributions and Watson Ladd, Hal Murray, Kurt Roeckx, and Ulrich Windl for their useful comments.

11. IANA Considerations

This memo includes no request to IANA.

12. Security Considerations

13. References

13.1. Normative References

[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>.

13.2. Informative References

[I-D.ietf-ntp-interleaved-modes]
Lichvar, M. and A. Malhotra, "NTP Interleaved Modes", draft-ietf-ntp-interleaved-modes-04 (work in progress), September 2020.

[IEEE1588]
Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2019, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems."", 11 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>.

Author's Address

Miroslav Lichvar
Red Hat
Purkynova 115
Brno 612 00
Czech Republic

Email: mlichvar@redhat.com

ntp
Internet-Draft
Intended status: Informational
Expires: 7 August 2021

R. Salz
Akamai Technologies

3 February 2021

The update registries draft
draft-rsalz-update-registries-02

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.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/richsalz/draft-rsalz-update-registries.

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 7 August 2021.

Copyright Notice

Table of Contents

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. 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.

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 are defined.

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 -- is 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 experimentation and development. Both registries are first-come first-served. The formal request to define the registries is in Section 16.

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

2.2. Extension Field Types

[RFC5906] mentioned the Extension Field Types registry, and defined it indirectly by defining 30 extensions (15 each for request and response) in Section 13. It did not provide a formal definition of the columns in the registry. Section 10 of [RFC5906] 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.

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

The following problems exists 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 document marks the erroneous values as reserved.

* Some values were mistakenly re-used.

2.3. Network Time Security Registries

[RFC8915] defines the Network Time Security (NTS) protocol. 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. New Registries

The following general guidelines apply to all registries defined here:

* Every entry reserves a partition for private use and experimentation.

* Registries with ASCII fields are now limited to uppercase letters; fields starting with 0x2D, the ASCII minus sign, are reserved for private use and experimentation.

* The policy for every registry is now specification required, as defined in Section 4.6 of [RFC8126].

The IESG is requested to choose three designated experts, with two being required to approve a registry change.

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

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 "-" 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 zero's. Each value must be an ASCII uppercase letter or minus sign

* 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 "-" 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 zero's. Each value must be an ASCII uppercase letter or minus sign.

* 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 should be [RFC5906] added, if possible.

The following Note is 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 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.

Field Type	Meaning	Reference
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	NTS Cookie	RFC 8915, Section 5.4
0x0204	Autokey Message Request	RFC 5906
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
0x2005	Reserved for historic reasons	This RFC
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
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
0x0902	Reserved for historic reasons	This RFC
0x8902	Reserved for historic reasons	This RFC
0xC902	Reserved for historic reasons	This RFC

Table 1

4.4. Network Time Security Key Establishment Record Types

The registration procedure is changed to specification required.

The following note should be added:

* Record Type numbers in the range 0x4000 through 0x7FFF, inclusive, are reserved for experimentation and development. IANA cannot assign them.

The existing entries are left unchanged.

4.5. Network Time Security Next Protocols

The registration procedure is changed to specification required.

The following note should be added:

* Protocol ID numbers in the range 0x8000 through 0xFFFF, inclusive, are reserved for experimentation and development. IANA cannot assign them.

The existing entries are left unchanged.

4.6. Network Time Security Error Codes

The registration procedure is changed to specification required.

The following note should be added:

* Error code numbers in the range 0x8000 through 0xFFFF, inclusive, are reserved for experimentation and development. IANA cannot assign them.

The existing entries are left unchanged.

4.7. Network Time Security Warning Codes

The registration procedure is changed to specification required.

The following note should be added:

* Warning code numbers in the range 0x8000 through 0xFFFF, inclusive, are reserved for experimentation and development. IANA cannot assign them.

The existing entries are left unchanged.

5. Acknowledgements

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

* Miroslav Lichvar, RedHat

* Daniel Franke, Akamai Technologies

* Danny Mayer, Network Time Foundation

* Michelle Cotton, IANA

And thanks to Harlen Stenn for providing popcorn.

6. 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/info/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/info/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/info/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/info/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/info/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/info/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/info/rfc8915>.

Author's Address

Rich Salz
Akamai Technologies

Email: rsalz@akamai.com

Calendaring Extensions Working Group
Internet-Draft
Obsoletes: 3339 (if approved)
Intended status: Standards Track
Expires: 26 July 2021

U. Sharma
Igalia, S.L.

22 January 2021

Date and Time on the Internet: Timestamps with additional information
draft-ryzokuken-datetime-extended-01

Abstract

This document defines a date and time format for use in Internet protocols for representation of dates and times using the proleptic Gregorian calendar, with optional extensions representing additional information including a time zone.

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 26 July 2021.

Copyright Notice

Table of Contents

1. Introduction

Date and time formats cause a lot of confusion and interoperability problems on the Internet. This document addresses many of the problems encountered and makes recommendations to improve consistency and interoperability when representing and using date and time in Internet protocols.

This document includes an extension to an Internet profile of the [ISO8601] standard for representation of dates and times using the proleptic Gregorian calendar alongside any additional information.

There are many ways in which date and time values might appear in Internet protocols: this document focuses on just one common usage, viz. timestamps for Internet protocol events. This limited consideration has the following consequences:

* All dates and times are assumed to be in the "current era", somewhere between 0000AD and 9999AD.

* All times expressed have a stated relationship (offset) to Coordinated Universal Time (UTC). Certain applications require the presence of a time zone in order to perform scheduling as well as handle Daylight Savings Time transitions properly. In that case, an optional time zone ID may be included.

* Timestamps can express times that occurred before the introduction of UTC. Such timestamps are expressed relative to universal time, using the best available practice at the stated time.

* Date and time expressions indicate an instant in time. Description of time periods, or intervals, is not covered here.

2. Definitions

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 [RFC2119].

UTC Coordinated Universal Time as maintained by the Bureau International des Poids et Mesures (BIPM).

second A basic unit of measurement of time in the International System of Units. It is defined as the duration of 9,192,631,770 cycles of microwave light absorbed or emitted by the hyperfine transition of cesium-133 atoms in their ground state undisturbed by external fields.

minute A period of time of 60 seconds. However, see also the restrictions in section Section 5.9 and Appendix C for how leap seconds are denoted within minutes.

hour A period of time of 60 minutes.

day A period of time of 24 hours.

leap year In the proleptic Gregorian calendar, a year which has 366 days. A leap year is a year whose number is divisible by four an integral number of times, except that if it is a centennial year (i.e. divisible by one hundred) it shall also be divisible by four hundred an integral number of times.

ABNF Augmented Backus-Naur Form, a format used to represent permissible strings in a protocol or language, as defined in [RFC2234].

Email Date/Time Format The date/time format used by Internet Mail as defined by [RFC2822].

Internet Date/Time Format The date/time format defined in section 5 of this document.

Timestamp This term is used in this document to refer to an unambiguous representation of some instant in time.

Z A suffix which, when applied to a time, denotes a UTC offset of 00:00; often spoken "Zulu" from the ICAO phonetic alphabet representation of the letter "Z".

Time Zone A time zone that is a included in the Time Zone Database (often called "tz" or "zoneinfo") maintained by IANA.

For more information about time scales, see Appendix E of [RFC1305], Section 3 of [ISO8601], and the appropriate ITU documents (ITU-R-TF).

3. Two Digit Years

The use of 2 (and 3) digit years was allowed but deprecated in [RFC3339], the predecessor of this document.

The use of such a format is no longer allowed, and implementations should use either a standard 4-digit year or the extended 6-digit value with a sign.

4. Local Time

4.1. Coordinated Universal Time (UTC)

Because the daylight saving rules for local time zones are so convoluted and can change based on local law at unpredictable times, true interoperability is best achieved by using Coordinated Universal Time (UTC). This specification by itself does not cater to local time zone rules. However, certain implementations may be expected to. For these situations, a timestamp may additionally include a local time zone that the implementations can take into account.

4.2. Local Offsets

The offset between local time and UTC is often useful information. For example, in electronic mail (RFC2822, [RFC2822]) the local offset provides a useful heuristic to determine the probability of a prompt response. Attempts to label local offsets with alphabetic strings have resulted in poor interoperability in the past [RFC1123]. As a result, RFC2822 [RFC2822] has made numeric offsets mandatory.

Numeric offsets are calculated as "local time minus UTC". So the equivalent time in UTC can be determined by subtracting the offset from the local time. For example, "18:50:00-04:00" is the same time as "22:50:00Z". (This example shows negative offsets handled by adding the absolute value of the offset.)

Numeric offsets may differ from UTC by any number of seconds, or even a fraction of seconds. This can be easily represented by including an optional seconds value in the offset, which may further optionally include a fraction of seconds behind a decimal point, for example "+12:34:56.789". This is especially useful in the case of certain historical time zones.

4.3. Unknown Local Offset Convention

If the time in UTC is known, but the offset to local time is unknown, this can be represented with an offset of "-00:00". This differs semantically from an offset of "Z" or "+00:00", which imply that UTC is the preferred reference point for the specified time. RFC2822 [RFC2822] describes a similar convention for email.

4.4. Unqualified Local Time

A number of devices currently connected to the Internet run their internal clocks in local time and are unaware of UTC. While the Internet does have a tradition of accepting reality when creating specifications, this should not be done at the expense of interoperability. Since interpretation of an unqualified local time zone will fail in approximately 23/24 of the globe, the interoperability problems of unqualified local time are deemed unacceptable for the Internet. Systems that are configured with a local time, are unaware of the corresponding UTC offset, and depend on time synchronization with other Internet systems, MUST use a mechanism that ensures correct synchronization with UTC. Some suitable mechanisms are:

* Use Network Time Protocol [RFC1305] to obtain the time in UTC.

* Use another host in the same local time zone as a gateway to the Internet. This host MUST correct unqualified local times that are transmitted to other hosts.

* Prompt the user for the local time zone and daylight saving rule settings.

5. Date and Time format

This section discusses desirable qualities of date and time formats and defines a format that extends the profile of ISO 8601 for use in Internet protocols.

5.1. Ordering

If date and time components are ordered from least precise to most precise, then a useful property is achieved. Assuming that the time zones of the dates and times are the same (e.g., all in UTC), expressed using the same string (e.g., all "Z" or all "+00:00"), all times have the same number of fractional second digits, and they all have the same suffix (or none), then the date and time strings may be sorted as strings (e.g., using the "strcmp()" function in C) and a time-ordered sequence will result. The presence of optional punctuation would violate this characteristic.

5.2. Human Readability

Human readability has proved to be a valuable feature of Internet protocols. Human readable protocols greatly reduce the costs of debugging since telnet often suffices as a test client and network analyzers need not be modified with knowledge of the protocol. On the other hand, human readability sometimes results in interoperability problems. For example, the date format "10/11/1996" is completely unsuitable for global interchange because it is interpreted differently in different countries. In addition, the date format in (RFC822) has resulted in interoperability problems when people assumed any text string was permitted and translated the three letter abbreviations to other languages or substituted date formats which were easier to generate (e.g. the format used by the C function "ctime"). For this reason, a balance must be struck between human readability and interoperability.

Because no date and time format is readable according to the conventions of all countries, Internet clients SHOULD be prepared to transform dates into a display format suitable for the locality. This may include translating UTC to local time as well as converting from the Gregorian calendar to the viewer’s preferred calendar.

5.3. Rarely Used Options

A format which includes rarely used options is likely to cause interoperability problems. This is because rarely used options are less likely to be used in alpha or beta testing, so bugs in parsing are less likely to be discovered. Rarely used options should be made mandatory or omitted for the sake of interoperability whenever possible.

5.4. Redundant Information

If a date/time format includes redundant information, that introduces the possibility that the redundant information will not correlate. For example, including the day of the week in a date/time format introduces the possibility that the day of week is incorrect but the date is correct, or vice versa. Since it is not difficult to compute the day of week from a date (see Appendix A), the day of week should not be included in a date/time format.

5.5. Simplicity

The complete set of date and time formats specified in ISO 8601 [ISO8601] is quite complex in an attempt to provide multiple representations and partial representations. Internet protocols have somewhat different requirements and simplicity has proved to be an important characteristic. In addition, Internet protocols usually need complete specification of data in order to achieve true interoperability. Therefore, the complete grammar for ISO 8601 is deemed too complex for most Internet protocols.

The following section defines a format that in an extension of a profile of ISO 8601 for use on the Internet. It is a conformant subset of the ISO 8601 extended format with additional information optionally suffixed. Simplicity is achieved by making most fields and punctuation mandatory.

5.6. Informative

The format should allow implementations to specify additional important information in addition to the bare timestamp. This is done by allowing implementations to include an informative suffix at the end with as many tags as required, each with a hyphen separated key and value. The value can be a hyphen delimited list of multiple values.

In case a key is repeated or conflicted, the implementations should give precedence to whichever value is positioned first.

5.7. Namespaced

Since the suffix can include all sorts of additional information, different standards bodies/organizations need a way to identify which part adheres to their standards. For this, all information needs to be namespaced. Each key is therefore divided into two hyphen-separated sections: the namespace and the key. For example, the calendar as defined by the Unicode consortium could be included as "u-ca-<value>".

All single-character namespaces are reserved for BCP47 extensions recorded in the BCP47 extensions registry. For these namespaces:

* Case differences are ignored.

* The namespace is restricted to single alphanum, corresponding to extension singletons ('x' can be used for a private use extension).

* In addition, for CLDR extensions:

  - There must be a "namespace-key" and it is restricted to 2 "alphanum" characters.

  - A "suffix-value" is limited to "3*8alphanum".

Multi-character namespaces can be registered specifically for use in this format. They are assigned by IANA using the "IETF Review" policy defined by [RFC5226]. This policy requires the development of an RFC, which SHALL define the name, purpose, processes, and procedures for maintaining the subtags. The maintaining or registering authority, including name, contact email, discussion list email, and URL location of the registry, MUST be indicated clearly in the RFC. The RFC MUST specify or include each of the following:

* The specification MUST reference the specific version or revision of this document that governs its creation and MUST reference this section of this document.

* The specification and all keys defined by the specification MUST follow the ABNF and other rules for the formation of keys as defined in this document. In particular, it MUST specify that case is not significant and that keys MUST NOT exceed eight characters in length.

* The specification MUST specify a canonical representation.

* The specification of valid keys MUST be available over the Internet and at no cost.

* The specification MUST be in the public domain or available via a royalty-free license acceptable to the IETF and specified in the RFC.

* The specification MUST be versioned, and each version of the specification MUST be numbered, dated, and stable.

* The specification MUST be stable. That is, namespace keys, once defined by a specification, MUST NOT be retracted or change in meaning in any substantial way.

* The specification MUST include, in a separate section, the registration form reproduced in this section (below) to be used in registering the namespace upon publication as an RFC.

* IANA MUST be informed of changes to the contact information and URL for the specification.

IANA will maintain a registry of allocated multi-character namespaces. This registry MUST use the record-jar format described by the ABNF in [RFC5646]. Upon publication of a namespace as an RFC, the maintaining authority defined in the RFC MUST forward this registration form to <mailto:iesg@ietf.org>, who MUST forward the request to <mailto:iana@iana.org>. The maintaining authority of the namespace MUST maintain the accuracy of the record by sending an updated full copy of the record to <mailto:iana@iana.org> with the subject line "TIMESTAMP FORMAT NAMESPACE UPDATE" whenever content changes. Only the 'Comments', 'Contact_Email', 'Mailing_List', and 'URL' fields MAY be modified in these updates.

Failure to maintain this record, maintain the corresponding registry, or meet other conditions imposed by this section of this document MAY be appealed to the IESG [RFC2028] under the same rules as other IETF decisions (see [RFC2026]) and MAY result in the authority to maintain the extension being withdrawn or reassigned by the IESG.

```
%%
Identifier:
Description:
Comments:
Added:
RFC:
Authority:
Contact_Email:
Mailing_List:
URL:
%%
```

Figure 1: Format of Records in the Timestamp Format Namespace Registry

'Identifier' contains the multi-character sequence assigned to the namespace. The Internet-Draft submitted to define the namespace SHOULD specify which sequence to use, although the IESG MAY change the assignment when approving the RFC.

'Description' contains the name and description of the namespace.

'Comments' is an OPTIONAL field and MAY contain a broader description of the namespace.

'Added' contains the date the namespace's RFC was published in the "date-full" format specified in Figure 2. For example: 2004-06-28 represents June 28, 2004, in the Gregorian calendar.

'RFC' contains the RFC number assigned to the namespace.

'Authority' contains the name of the maintaining authority for the namespace.

'Contact_Email' contains the email address used to contact the maintaining authority.

'Mailing_List' contains the URL or subscription email address of the mailing list used by the maintaining authority.

'URL' contains the URL of the registry for this namespace.

The determination of whether an Internet-Draft meets the above conditions and the decision to grant or withhold such authority rests solely with the IESG and is subject to the normal review and appeals process associated with the RFC process.

5.8. Internet Date/Time Format

The following extension of a profile of [ISO8601] dates SHOULD be used in new protocols on the Internet. This is specified using the syntax description notation defined in [RFC2234].

```
alphanum       = ALPHA / DIGIT

date-year      = 4DIGIT / ("+" / "-") 6DIGIT
date-month     = 2DIGIT ; 01-12
date-mday      = 2DIGIT ; 01-28, 01-29, 01-30, 01-31 based on month/year
date-full      = date-year "-" date-month "-" date-mday

time-hour      = 2DIGIT ; 00-23
time-minute    = 2DIGIT ; 00-59
time-second    = 2DIGIT ; 00-58, 00-59, 00-60 based on leap second rules
time-secfrac   = "." 1*DIGIT
time-partial   = time-hour ":" time-minute ":" time-second [time-secfrac]
time-numoffset = ("+" / "-") time-partial
time-offset    = "Z" / time-numoffset
time-full      = time-partial time-offset

time-zone-char = ALPHA / "." / "_"
time-zone-part = time-zone-char *13(time-zone-char / DIGIT / "-" / "+") ; but
not "." or ".."
time-zone-id   = time-zone-part *("/" time-zone-part)
time-zone      = "[" time-zone-id "]"

namespace      = 1*alphanum
namespace-key  = 1*alphanum
suffix-key     = namespace ["-" namespace-key]

suffix-value   = 1*alphanum
suffix-values  = suffix-value *("-" suffix-value)
suffix-tag     = "[" suffix-key "-" suffix-values "]"
suffix         = [timezone] *suffix-tag

date-time      = date-full "T" time-full suffix
```

Figure 2

> NOTE 1: Per [RFC2234] and ISO8601, the "T" and "Z" characters in this syntax may alternatively be lower case "t" or "z" respectively.

This date/time format may be used in some environments or contexts that distinguish between the upper- and lower-case letters 'A'-'Z' and 'a'-'z' (e.g. XML). Specifications that use this format in such environments MAY further limit the date/time syntax so that the

letters 'T' and 'Z' used in the date/time syntax must always be upper case. Applications that generate this format SHOULD use upper case letters.

> NOTE 2: ISO 8601 defines date and time separated by "T". Applications using this syntax may choose, for the sake of readability, to specify a full-date and full-time separated by (say) a space character.

5.9. Restrictions

The grammar element date-mday represents the day number within the current month. The maximum value varies based on the month and year as follows:

Month Number	Month/Year	Maximum value of date-mday
01	January	31
02	February, normal	28
02	February, leap year	29
03	March	31
04	April	30
05	May	31
06	June	30
07	July	31
08	August	31
09	September	30
10	October	31
11	November	30
12	December	31

Table 1: Days in each month

Appendix B contains sample C code to determine if a year is a leap year.

The grammar element time-second may have the value "60" at the end of months in which a leap second occurs - to date: June (XXXX-06-30T23:59:60Z) or December (XXXX-12-31T23:59:60Z); see Appendix C for a table of leap seconds. It is also possible for a leap second to be subtracted, at which times the maximum value of time-second is "58". At all other times the maximum value of time-second is "59". Further, in time zones other than "Z", the leap second point is shifted by the zone offset (so it happens at the same instant around the globe).

Leap seconds cannot be predicted far into the future. The International Earth Rotation Service publishes bulletins (IERS) that announce leap seconds with a few weeks' warning. Applications should not generate timestamps involving inserted leap seconds until after the leap seconds are announced.

Although ISO 8601 permits the hour to be "24", this extension of a profile of ISO 8601 only allows values between "00" and "23" for the hour in order to reduce confusion.

5.10. Examples

Here are some examples of Internet date/time format.

```
1985-04-12T23:20:50.52Z
```

Figure 3

This represents 20 minutes and 50.52 seconds after the 23rd hour of April 12th, 1985 in UTC.

```
+001985-04-12T23:20:50.52Z
```

Figure 4

This represents the same instant as the previous example but with the expanded 6-digit year format.

```
1996-12-19T16:39:57-08:00
```

Figure 5

This represents 39 minutes and 57 seconds after the 16th hour of December 19th, 1996 with an offset of -08:00 from UTC (Pacific Standard Time). Note that this is equivalent to 1996-12-20T00:39:57Z in UTC.

```
1996-12-19T16:39:57-08:00[America/Los_Angeles]
```

Figure 6

This represents the exact same instant as the previous example but additionally specifies the human time zone associated with it for time zone aware implementations to take into account.

```
1996-12-19T16:39:57-08:00[America/Los_Angeles][u-ca-hebrew]
```

Figure 7

This represents the exact same instant but it informs calendar-aware implementations that they should project it to the Hebrew calendar.

```
1990-12-31T23:59:60Z
```

Figure 8

This represents the leap second inserted at the end of 1990.

```
1990-12-31T15:59:60-08:00
```

Figure 9

This represents the same leap second in Pacific Standard Time, 8 hours behind UTC.

```
1937-01-01T12:00:27.87+00:19:32.130
```

Figure 10

This represents the same instant of time as noon, January 1, 1937, Netherlands time. Standard time in the Netherlands was exactly 19 minutes and 32.13 seconds ahead of UTC by law from 1909-05-01 through 1937-06-30.

```
1937-01-01T12:00:27.87+00:19:32.130[u-ca-japanese]
```

Figure 11

This represents the exact same instant as the previous example but additionally specifies the human calendar associated with it for calendar aware implementations to take into account.

```
1937-01-01T12:00:27.87+00:19:32.130[u-ca-islamic-civil]
```

Figure 12

Since there's not a single agreed upon way to deal with dates in the Islamic calendar, it provides another value to disambiguate between the different interpretations.

```
1937-01-01T12:00:27.87+00:19:32.130[x-foo-bar][x-baz-bat]
```

Figure 13

This timestamp utilizes the private use namespace to declare two additional pieces of information in the suffix that can be interpreted by any compatible implementations and ignored otherwise.

6. Security Considerations

Since the local time zone of a site may be useful for determining a time when systems are less likely to be monitored and might be more susceptible to a security probe, some sites may wish to emit times in UTC only. Others might consider this to be loss of useful functionality at the hands of paranoia.

7. Normative references

[RFC2822] Resnick, P., Ed., "Internet Message Format", IETF RFC 2822, IETF RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/info/rfc2822>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", IETF RFC 2234, IETF RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/info/rfc2234>.

[RFC1123] Braden, R., Ed., "Requirements for Internet HostsApplication and Support", IETF RFC 1123, IETF RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", IETF RFC 1305, IETF RFC 1305, DOI 10.17487/RFC1305, March 1992, <https://www.rfc-editor.org/info/rfc1305>.

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

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", IETF RFC 5646, IETF RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC2026] Bradner, S., "The Internet Standards ProcessRevision 3", IETF RFC 2026, IETF RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2028] Hovey, R. and S. Bradner, "The Organizations Involved in the IETF Standards Process", IETF RFC 2028, IETF RFC 2028, DOI 10.17487/RFC2028, October 1996, <https://www.rfc-editor.org/info/rfc2028>.

8. Bibliography

[ISO8601] International Organization for Standardization, "Data elements and interchange formats", ISO 8601:1988, June 1988, <https://www.iso.org/standard/15903.html>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", IETF RFC 3339, IETF RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

Appendix A. Day of the Week

The following is a sample C subroutine loosely based on Zeller's Congruence (ZELLER) which may be used to obtain the day of the week for dates on or after 0000-03-01:

```
char *day_of_week(int day, int month, int year)
{
    int cent;
    char *dayofweek[] = {
        "Sunday", "Monday", "Tuesday", "Wednesday",
        "Thursday", "Friday", "Saturday"
    };

    /* adjust months so February is the last one */
    month -= 2;
    if (month < 1) {
        month += 12;
        --year;
    }
    /* split by century */
    cent = year / 100;
    year %= 100;
    return (dayofweek[((26 * month - 2) / 10 + day + year
                    + year / 4 + cent / 4 + 5 * cent) % 7]);
}
```

Figure 14

Appendix B. Leap Years

Here is a sample C subroutine to calculate if a year is a leap year:

```
/* This returns non-zero if year is a leap year.  Must use 4 digit
    year.
*/
int leap_year(int year)
{
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
```

Figure 15

Appendix C. Leap Seconds

Information about leap seconds can be found at the US Navy Oceanography Portal (https://www.usno.navy.mil/USNO/time/master-clock/leap-seconds). In particular, it notes that:

> The decision to introduce a leap second in UTC is the responsibility of the International Earth Rotation Service (IERS). According to the CCIR Recommendation, first preference is given to the opportunities at the end of December and June, and second preference to those at the end of March and September.

When required, insertion of a leap second occurs as an extra second at the end of a day in UTC, represented by a timestamp of the form YYYY-MM-DDT23:59:60Z. A leap second occurs simultaneously in all time zones, so that time zone relationships are not affected. See section Section 5.10 for some examples of leap second times.

The following table is an excerpt from the table maintained by the United States Naval Observatory. The source data is located at the US Navy Oceanography Portal (ftp://maia.usno.navy.mil/ser7/tai-utc.dat).

This table shows the date of the leap second, and the difference between the time standard TAI (which isn't adjusted by leap seconds) and UTC after that leap second.

UTC Date	TAI - UTC After Leap Second
1972-06-30	11
1972-12-31	12
1973-12-31	13
1974-12-31	14
1975-12-31	15
1976-12-31	16
1977-12-31	17
1978-12-31	18
1979-12-31	19
1981-06-30	20
1982-06-30	21
1983-06-30	22
1985-06-30	23
1987-12-31	24
1989-12-31	25

1990-12-31	26
1992-06-30	27
1993-06-30	28
1994-06-30	29
1995-12-31	30
1997-06-30	31
1998-12-31	32

Table 2: Historic leap seconds

Author's Address

Ujjwal Sharma
Igalia, S.L.

Email: ryzokuken@igalia.com

Calendaring Extensions Working Group
Internet-Draft
Obsoletes: 3339 (if approved)
Intended status: Standards Track
Expires: 25 August 2021

U. Sharma
Igalia, S.L.

21 February 2021

Date and Time on the Internet: Timestamps
draft-ryzokuken-datetime-updated-00

Abstract

This document defines a date and time format for use in Internet protocols that is a profile of the ISO 8601 standard for representation of dates and times using the proleptic Gregorian calendar.

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 25 August 2021.

Copyright Notice

Table of Contents

1. Introduction

Date and time formats cause a lot of confusion and interoperability problems on the Internet. This document addresses many of the problems encountered and makes recommendations to improve consistency and interoperability when representing and using date and time in Internet protocols.

This document includes an Internet profile of the [ISO8601] standard for representation of dates and times using the proleptic Gregorian calendar.

There are many ways in which date and time values might appear in Internet protocols: this document focuses on just one common usage, viz. timestamps for Internet protocol events. This limited consideration has the following consequences:

* All dates and times are assumed to be in the "current era", somewhere between 0000AD and 9999AD.

* All times expressed have a stated relationship (offset) to Coordinated Universal Time (UTC). (This is distinct from some usage in scheduling applications where a local time and location may be known, but the actual relationship to UTC may be dependent on the unknown or unknowable actions of politicians or administrators. The UTC time corresponding to 17:00 on 23rd March 2005 in New York may depend on administrative decisions about daylight savings time. This specification steers well clear of such considerations.)

* Timestamps can express times that occurred before the introduction of UTC. Such timestamps are expressed relative to universal time, using the best available practice at the stated time.

* Date and time expressions indicate an instant in time. Description of time periods, or intervals, is not covered here.

2. Definitions

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 [RFC2119].

UTC Coordinated Universal Time as maintained since 1988 by the Bureau International des Poids et Mesures (BIPM) in conjunction with leap seconds as announced by the International Earth Rotation and Reference Frames Service [IERS]. From 1972 through 1987 UTC was maintained entirely by Bureau International de l'Heure (BIH). Before 1972 UTC was not generally recognized and civil time was determined by individual jurisdictions using different techniques for attempting to follow Universal Time based on measuring the rotation of the earth.

second The unit of time in the International System of Units. Since Resolution 1 of the 13th CGPM on 1967-10-13 [CGPM] the second is defined as the duration of 9,192,631,770 cycles of microwave radiation absorbed or emitted by the hyperfine transition of cesium-133 atoms in their ground state undisturbed by external fields, but this definition was not in practical use for civil time until 1972-01-01. Prior to 1972-01-01 civil time was based on Universal Time which was measured by observations of the rotation of the earth, and the practical definition of the second was 1/86400 of the mean solar day.

minute A period of time of 60 seconds. However, see also the restrictions in section Section 5.7 and Appendix C for how leap seconds are denoted within minutes.

hour A period of time of 60 minutes.

day Starting 1972-01-01 a duration of 86400 SI seconds for the UTC time scale. In other contexts the duration of one mean solar day as agreed internationally by the 1884 International Meridian Conference and measured using Universal Time.

leap year In the proleptic Gregorian calendar, a year which has 366 days. A leap year is a year whose number is divisible by four an integral number of times, except that if it is a centennial year (i.e. divisible by one hundred) it shall also be divisible by four hundred an integral number of times.

ABNF Augmented Backus-Naur Form, a format used to represent permissible strings in a protocol or language, as defined in [RFC2234].

Email Date/Time Format The date/time format used by Internet Mail as defined by [RFC2822].

Internet Date/Time Format The date/time format defined in section 5 of this document.

Timestamp This term is used in this document to refer to an unambiguous representation of some instant in time.

Z A suffix which, when applied to a time, denotes a UTC offset of 00:00; often spoken "Zulu" from the ICAO phonetic alphabet representation of the letter "Z".

For more information about time scales, see Appendix E of [RFC1305], Section 3 of [ISO8601], and the appropriate ITU documents [ITU-R-TF].

3. Two Digit Years

The use of 2 (and 3) digit years was allowed but deprecated in [RFC3339], the predecessor of this document.

The use of such a format is no longer allowed, and implementations should use either a standard 4-digit year or the extended 6-digit value with a sign.

4. Local Time

4.1. Coordinated Universal Time (UTC)

Because the daylight saving rules for local time zones are so convoluted and can change based on local law at unpredictable times, true interoperability is best achieved by using Coordinated Universal Time (UTC). This specification does not cater to local time zone rules.

4.2. Local Offsets

The offset between local time and UTC is often useful information. For example, in electronic mail ([RFC2822]) the local offset provides a useful heuristic to determine the probability of a prompt response. Attempts to label local offsets with alphabetic strings have resulted in poor interoperability in the past [RFC1123]. As a result, [RFC2822] has made numeric offsets mandatory.

Numeric offsets are calculated as "local time minus UTC". So the equivalent time in UTC can be determined by subtracting the offset from the local time. For example, "18:50:00-04:00" is the same time as "22:50:00Z". (This example shows negative offsets handled by adding the absolute value of the offset.)

Numeric offsets may differ from UTC by any number of seconds, or even a fraction of seconds. This can be easily represented by including an optional seconds value in the offset, which may further optionally include a fraction of seconds behind a decimal point, for example "+12:34:56.789". This is especially useful in the case of certain historical time zones.

4.3. Unknown Local Offset Convention

If the time in UTC is known, but the offset to local time is unknown, this can be represented with an offset of "-00:00". This differs semantically from an offset of "Z" or "+00:00", which imply that UTC is the preferred reference point for the specified time. RFC2822 [RFC2822] describes a similar convention for email.

4.4. Unqualified Local Time

A number of devices currently connected to the Internet run their internal clocks in local time and are unaware of UTC. While the Internet does have a tradition of accepting reality when creating specifications, this should not be done at the expense of interoperability. Since interpretation of an unqualified local time zone will fail in approximately 23/24 of the globe, the interoperability problems of unqualified local time are deemed unacceptable for the Internet. Systems that are configured with a

local time, are unaware of the corresponding UTC offset, and depend on time synchronization with other Internet systems, MUST use a mechanism that ensures correct synchronization with UTC. Some suitable mechanisms are:

* Use Network Time Protocol [RFC1305] to obtain the time in UTC.

* Use another host in the same local time zone as a gateway to the Internet. This host MUST correct unqualified local times that are transmitted to other hosts.

* Prompt the user for the local time zone and daylight saving rule settings.

5. Date and Time format

This section discusses desirable qualities of date and time formats and defines a profile of ISO 8601 for use in Internet protocols.

5.1. Ordering

If date and time components are ordered from least precise to most precise, then a useful property is achieved. Assuming that the time zones of the dates and times are the same (e.g., all in UTC), expressed using the same string (e.g., all "Z" or all "+00:00"), and all times have the same number of fractional second digits then the date and time strings may be sorted as strings (e.g., using the "strcmp()" function in C) and a time-ordered sequence will result. The presence of optional punctuation would violate this characteristic.

5.2. Human Readability

Human readability has proved to be a valuable feature of Internet protocols. Human readable protocols greatly reduce the costs of debugging since telnet often suffices as a test client and network analyzers need not be modified with knowledge of the protocol. On the other hand, human readability sometimes results in interoperability problems. For example, the date format "10/11/1996" is completely unsuitable for global interchange because it is interpreted differently in different countries. In addition, the date format in (RFC822) has resulted in interoperability problems when people assumed any text string was permitted and translated the three letter abbreviations to other languages or substituted date formats which were easier to generate (e.g. the format used by the C function "ctime"). For this reason, a balance must be struck between human readability and interoperability.

Because no date and time format is readable according to the conventions of all countries, Internet clients SHOULD be prepared to transform dates into a display format suitable for the locality. This may include translating UTC to local time.

5.3. Rarely Used Options

A format which includes rarely used options is likely to cause interoperability problems. This is because rarely used options are less likely to be used in alpha or beta testing, so bugs in parsing are less likely to be discovered. Rarely used options should be made mandatory or omitted for the sake of interoperability whenever possible.

5.4. Redundant Information

If a date/time format includes redundant information, that introduces the possibility that the redundant information will not correlate. For example, including the day of the week in a date/time format introduces the possibility that the day of week is incorrect but the date is correct, or vice versa. Since it is not difficult to compute the day of week from a date (see Appendix A), the day of week should not be included in a date/time format.

5.5. Simplicity

The complete set of date and time formats specified in ISO 8601 [ISO8601] is quite complex in an attempt to provide multiple representations and partial representations. Internet protocols have somewhat different requirements and simplicity has proved to be an important characteristic. In addition, Internet protocols usually need complete specification of data in order to achieve true interoperability. Therefore, the complete grammar for ISO 8601 is deemed too complex for most Internet protocols.

The following section defines a profile of ISO 8601 for use on the Internet. It is a conformant subset of the ISO 8601 extended format. Simplicity is achieved by making most fields and punctuation mandatory.

5.6. Internet Date/Time Format

The following profile of [ISO8601] dates SHOULD be used in new protocols on the Internet. This is specified using the syntax description notation defined in [RFC2234].

```
date-year      = 4DIGIT / ("+" / "-") 6DIGIT
date-month     = 2DIGIT ; 01-12
date-mday      = 2DIGIT ; 01-28, 01-29, 01-30, 01-31 based on month/year
date-full      = date-year "-" date-month "-" date-mday

time-hour      = 2DIGIT ; 00-23
time-minute    = 2DIGIT ; 00-59
time-second    = 2DIGIT ; 00-58, 00-59, 00-60 based on leap second rules
time-secfrac   = "." 1*DIGIT
time-partial   = time-hour ":" time-minute ":" time-second [time-secfrac]
time-numoffset = ("+" / "-") time-partial
time-offset    = "Z" / time-numoffset
time-full      = time-partial time-offset

date-time      = date-full "T" time-full
```

Figure 1

> NOTE 1: Per [RFC2234] and ISO8601, the "T" and "Z" characters in this syntax may alternatively be lower case "t" or "z" respectively.

This date/time format may be used in some environments or contexts that distinguish between the upper- and lower-case letters 'A'-'Z' and 'a'-'z' (e.g. XML). Specifications that use this format in such environments MAY further limit the date/time syntax so that the letters 'T' and 'Z' used in the date/time syntax must always be upper case. Applications that generate this format SHOULD use upper case letters.

> NOTE 2: ISO 8601 defines date and time separated by "T". Applications using this syntax may choose, for the sake of readability, to specify a full-date and full-time separated by (say) a space character.

5.7. Restrictions

The grammar element date-mday represents the day number within the current month. The maximum value varies based on the month and year as follows:

Month Number	Month/Year	Maximum value of date-mday
01	January	31
02	February, normal	28
02	February, leap year	29
03	March	31
04	April	30
05	May	31
06	June	30
07	July	31
08	August	31
09	September	30
10	October	31
11	November	30
12	December	31

Table 1: Days in each month

Appendix B contains sample C code to determine if a year is a leap year.

The grammar element time-second may have the value "60" at the end of months in which a leap second occurs - to date: June (XXXX-06-30T23:59:60Z) or December (XXXX-12-31T23:59:60Z); see Appendix C for a table of leap seconds. It is also possible for a leap second to be subtracted, at which times the maximum value of time-second is "58". At all other times the maximum value of time-second is "59". Further, in time zones other than "Z", the leap second point is shifted by the zone offset (so it happens at the same instant around the globe).

Leap seconds cannot be predicted far into the future. The International Earth Rotation Service publishes bulletins [IERS] that announce leap seconds with a few weeks' warning. Applications should not generate timestamps involving inserted leap seconds until after the leap seconds are announced.

Although ISO 8601 permits the hour to be "24", this profile of ISO 8601 only allows values between "00" and "23" for the hour in order to reduce confusion.

5.8. Examples

Here are some examples of Internet date/time format.

```
1985-04-12T23:20:50.52Z
```

Figure 2

This represents 20 minutes and 50.52 seconds after the 23rd hour of April 12th, 1985 in UTC.

```
+001985-04-12T23:20:50.52Z
```

Figure 3

This represents the same instant as the previous example but with the expanded 6-digit year format.

```
1996-12-19T16:39:57-08:00
```

Figure 4

This represents 39 minutes and 57 seconds after the 16th hour of December 19th, 1996 with an offset of -08:00 from UTC (Pacific Standard Time). Note that this is equivalent to 1996-12-20T00:39:57Z in UTC.

```
1990-12-31T23:59:60Z
```

Figure 5

This represents the leap second inserted at the end of 1990.

```
1990-12-31T15:59:60-08:00
```

Figure 6

This represents the same leap second in Pacific Standard Time, 8 hours behind UTC.

1937-01-01T12:00:27.87+00:19:32.130

Figure 7

This represents the same instant of time as noon, January 1, 1937, Netherlands time. Standard time in the Netherlands was exactly 19 minutes and 32.13 seconds ahead of UTC by law from 1909-05-01 through 1937-06-30.

6. Security Considerations

Since the local time zone of a site may be useful for determining a time when systems are less likely to be monitored and might be more susceptible to a security probe, some sites may wish to emit times in UTC only. Others might consider this to be loss of useful functionality at the hands of paranoia.

7. Normative references

[RFC2822] Resnick, P., Ed., "Internet Message Format", IETF RFC 2822, IETF RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/info/rfc2822>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", IETF RFC 2234, IETF RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/info/rfc2234>.

[RFC1123] Braden, R., Ed., "Requirements for Internet HostsApplication and Support", IETF RFC 1123, IETF RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", IETF RFC 1305, IETF RFC 1305, DOI 10.17487/RFC1305, March 1992, <https://www.rfc-editor.org/info/rfc1305>.

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

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", IETF RFC 5646, IETF RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC2026] Bradner, S., "The Internet Standards ProcessRevision 3", IETF RFC 2026, IETF RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2028] Hovey, R. and S. Bradner, "The Organizations Involved in the IETF Standards Process", IETF RFC 2028, IETF RFC 2028, DOI 10.17487/RFC2028, October 1996, <https://www.rfc-editor.org/info/rfc2028>.

8. Bibliography

[ISO8601] International Organization for Standardization, "Data elements and interchange formats", ISO 8601:1988, June 1988, <https://www.iso.org/standard/15903.html>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", IETF RFC 3339, IETF RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[ISO8601-2000] International Organization for Standardization, "Data elements and interchange formats", ISO 8601:2000, December 2000, <https://www.iso.org/standard/26780.html>.

[ITU-R-TF] "", ITU-R TF.460-6.

[CGPM] "Resolution 1 of the 13th CGPM, 1967".

[ZELLER] "Zeller, Chr. Kalender-Formeln. Acta Math. 9 (1887), 131136. doi:10.1007/BF02406733".

[IERS] "International Earth Rotation Service Bulletins".

Appendix A. Day of the Week

The following is a sample C subroutine loosely based on Zeller's Congruence [ZELLER] which may be used to obtain the day of the week for dates on or after 0000-03-01:

```
char *day_of_week(int day, int month, int year)
{
    int cent;
    char *dayofweek[] = {
        "Sunday", "Monday", "Tuesday", "Wednesday",
        "Thursday", "Friday", "Saturday"
    };

    /* adjust months so February is the last one */
    month -= 2;
    if (month < 1) {
        month += 12;
        --year;
    }
    /* split by century */
    cent = year / 100;
    year %= 100;
    return (dayofweek[((26 * month - 2) / 10 + day + year
                    + year / 4 + cent / 4 + 5 * cent) % 7]);
}
```

Figure 8

Appendix B. Leap Years

Here is a sample C subroutine to calculate if a year is a leap year:

```
/* This returns non-zero if year is a leap year.  Must use 4 digit
    year.
*/
int leap_year(int year)
{
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
```

Figure 9

Appendix C. Leap Seconds

In 1970 CCIR Recommendation 460 produced international agreement that starting on 1972-01-01 radio broadcast time signals should provide SI seconds with occasional leaps of 1 SI second as necessary to agree with Universal Time. The time scale in radio broadcasts became known as UTC, and the current version of that recommendation is [ITU-R-TF]. Since 1988 IERS has the responsibility for announcing when leap seconds will be introduced into UTC (https://www.iers.org/SharedDocs/Publikationen/EN/IERS/Documents/IERS_Leap_Seconds.pdf?__blob=publicationFile&v=1). Further

information about leap seconds can be found at the US Navy Oceanography Portal (https://www.usno.navy.mil/USNO/time/master-clock/leap-seconds). In particular, it notes that:

> The decision to introduce a leap second in UTC is the responsibility of the International Earth Rotation Service [IERS]. According to the CCIR Recommendation, first preference is given to the opportunities at the end of December and June, and second preference to those at the end of March and September.

When required, insertion of a leap second occurs as an extra second at the end of a day in UTC, represented by a timestamp of the form YYYY-MM-DDT23:59:60Z. A leap second occurs simultaneously in all time zones, so that time zone relationships are not affected. See section Section 5.8 for some examples of leap second times.

The following table is an excerpt from the table maintained by the IERS. The source data are located at the Earth Orientation Parameters Product Centre at Observatoire de Paris (https://hpiers.obspm.fr/eop-pc/index.php?index=TAI-UTC_tab&lang=en).

For dates after the initial adjustment on 1972-01-01 this table shows the date of the leap second, and the difference between the time scale TAI (which is not adjusted by leap seconds) and UTC after that leap second.

UTC Date	TAI - UTC After Leap Second
1972-06-30	11
1972-12-31	12
1973-12-31	13
1974-12-31	14
1975-12-31	15
1976-12-31	16
1977-12-31	17
1978-12-31	18
1979-12-31	19
1981-06-30	20

1982-06-30	21
1983-06-30	22
1985-06-30	23
1987-12-31	24
1989-12-31	25
1990-12-31	26
1992-06-30	27
1993-06-30	28
1994-06-30	29
1995-12-31	30
1997-06-30	31
1998-12-31	32
2005-12-31	33
2008-12-31	34
2012-06-30	35
2015-06-30	36
2016-12-31	37

Table 2: Historic leap seconds

Author's Address

Ujjwal Sharma
Igalia, S.L.

Email: ryzokuken@igalia.com