

TICTOC
Internet-Draft
Updates: none (if approved)
Intended status: Standards Track
Expires: 12 May 2022

J.I.A-H. Alvarez-Hamelin, Ed.
Universidad de Buenos Aires - CONICET
D. Samaniego
A.A. Ortega
Universidad de Buenos Aires
R. Geib
Deutsche Telekom
8 November 2021

Synchronizing Internet Clock frequency protocol (sic)
draft-alvarez-hamelin-tictoc-sic-08

Abstract

Synchronizing Internet Clock (sic) Frequency specifies a new secure method to synchronize clocks on the Internet, assuring smoothness (i.e., frequency stability) and robustness to man-in-the-middle attacks. This protocol is oriented to assure the quality of Internet performance measurements, where they are frequently obtained as the difference of timestamps, hence frequency stability is needed. In 90% of all cases, Synchronized Internet Clock Frequency is highly accurate, with a Maximum Time Interval Error of fewer than 25 microseconds by a minute. Synchronized Internet Clock Frequency is based on a regular packet exchange and works with commodity terminal hardware.

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 [RFC2119].

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 12 May 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. The sic frequency protocol overview	4
3. The formal definition of sic frequency protocol	8
3.1. Algorithm description	9
3.2. Protocol definitions	13
3.3. Protocol packet specification	15
3.4. Minimum sic deployment	16
4. Implementation of sic frequency protocol	17
4.1. Evaluation	17
5. Conclusions	19
6. Security Considerations	19
7. IANA Considerations	20
8. Acknowledgements	20
9. References	20
9.1. Normative References	20
9.2. Informative References	21
Appendix A. Example of RTT to NTP servers	22
Authors' Addresses	24

1. Introduction

One way metric measurements [RFC7679] require synchronization of sender and receiver clocks to obtain reliable results. This synchronization should be smooth (i.e., operate without steps in synchronization and detect and reach a stable operation state), offer a precise frequency, easily implemented in any host on the Internet, and faithful. A reliable clock frequency is needed to perform precise differential metric measurements between any two hosts to capture performance metrics like packet delay variation or loss of packets exchanged between arbitrary measurement hosts. The required

clock synchronization is designed to be implementable by software to allow deployment with hosts in arbitrary locations. It cannot replace or compete against commodity clock synchronization standards (and any intent to do so is out of scope). Finally, some security measures are needed to avoid some common types of attacks in the Internet, given more robustness to the system.

There are different types of clock synchronization on the Internet. NTP [RFC5905] remains one of the most popular because a potential user does not need any extra hardware, and it is practically a standard in most of the operating-systems distributions. Its working principle relies on time servers with precise clock source, as atomic clocks or GPS based. For most of the needs, NTP provides an accurate synchronization. Moreover, NTP recently incorporates some strategies oriented to avoid man-in-the-middle (MitM) attacks. NTP's potential accuracy is in the order of tens of milliseconds..

Another proposal is the TSClocks [ToN2008], which take advantage of the internal computers' clock. This work has been shown an attractive solution because it is not expensive and can be used on any computer connected to the Internet. This solution was proposed in the beginning at LAN (Local Area Network) level, and then it has been extended to other situations. In [ToN2008] authors report a differential clock error of about half of hundred of microseconds for a WAN connection with 40ms of RTT (Round Trip Time), i.e., the absolute error is the same order that RTT.

When accuracy and stability are needed, further options arise, e.g., the PTP clock [RFC8173] (this mechanism was also defined as the IEEE Std. 1588-2008). However, the PTP clock incorporates specialized hardware to provide a highly accurate clock required in each point to be synchronized. Also, the GPS (Global Position System) requires specialized hardware at every point of measurement. While GPS may be less expensive than PTP, the GPS unit requires a sky-clear view for working. The latter may be costly or impossible in some locations due to the antenna installation.

This document introduces the Synchronizing Internet Clock frequency (sic frequency), which is a protocol providing synchronized differential clocks (i.e., when the amount to measure is the elapsed time between two timestamps) in two endpoints connected to the Internet. While synchronized absolute clocks aim at a measurement of exact time differences between them, synchronized differential clocks allow measurements during identical time intervals at two locations. This property is useful for Internet performance measurements, like congestion, jitter, or delay variation; which are based on the elapsed time between timestamps.

The sic frequency design is close to TSClocks, but it takes advantage of statistics to perform better. The sic frequency synchronization relies on Internet-based delay measurements, including the frequent routes' change detection. Finally, our implementation also contemplates protecting MitM attacks, including light but a powerful signature of measurements in each packet. The sic frequency protocol does neither put constraints on the quality of a server's clock nor require a limitation of synchronized end systems' physical distance. The quality of the absolute synchronization is determined by the stability of the reference clock and the $RTT/2$ as in NTP, but the sic frequency protocol has a better performance in frequency stability than NTP.

Finally, we mention the [ITU-G.8260] shows a methodology to measure delays in networks. It is based on filtering that selects some packets to perform the delay computation. The packet selection is based on the minimum and average RTT, and we show that both of them have some statistical problems to determine both, the average and the minimum RTT (see Section 2).

2. The sic frequency protocol overview

Synchronizing Internet Clock frequency (sic frequency) is a protocol providing synchronized differential clocks in two endpoints connected to the Internet. Synchronized differential clocks allow measurements during identical time intervals at two locations. This is useful if congestion, packet loss or a variation in delay is to be measured. The model of typical Internet time-measurement is shown in Figure 1.

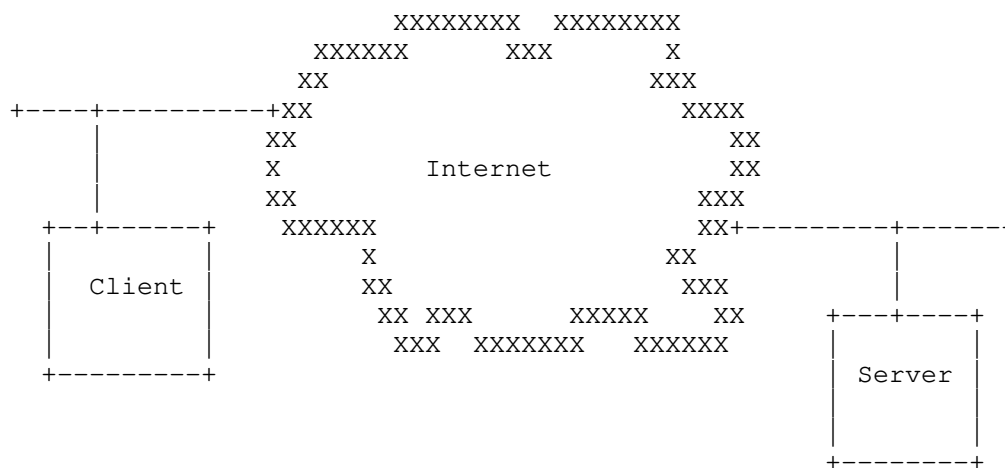


Figure 1: The clock synchronization of sic.--

In this model, sic frequency performs measurements with packets in the way shown in

Figure 2.

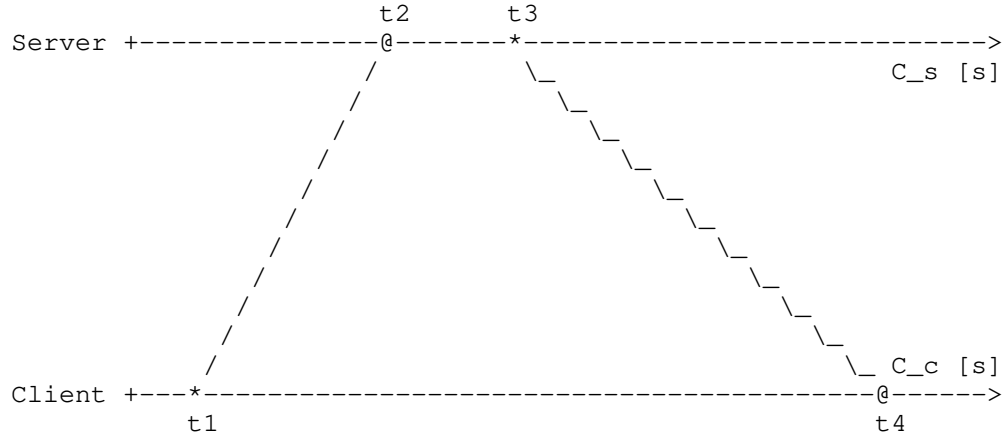


Figure 2: Time line of packets.--

Here, C_s is the server clock, C_c is the client clock and $t1...t4$ are timestamps.

Figure 2 shows a horizontal timeline for client and server. The diagonal lines depict a packet traversing some physical space (wires, routers, and switches). The packet travel times are not assumed to be identical because routes and background load may differ in each direction.

The difference between the client clock C_c and the server clock C_s can be modeled as:

$$C_c = C_s + \phi ,$$

$$\phi(t) = C_c(t) - C_s(t) , \quad (1)$$

where ϕ is the absolute clock difference. If RTT is constant (i.e. little or no background load) and routes are symmetric in both directions, the difference between clocks can be computed as:

$$\phi[c \rightarrow s] = t1 - (t2 - RTT/2) , \quad (2)$$

$$\phi[c \leftarrow s] = t4 - (t3 + RTT/2) , \quad (3)$$

and $\phi[c \rightarrow s] = \phi[c \leftarrow s]$. The general equation for the RTT is:

$$RTT = (t_2 - t_1) + (t_4 - t_3) . \quad (4)$$

Computing Equations 2 and 3 for this simplified case allows calculation of ϕ as an RTT function. Note that if routes are not symmetrical, it is impossible to determine the absolute clocks' difference.

The sic frequency protocol is based on statistics, background traffic and network behavior observations. The RTT between two endpoints follows a heavy-tailed distribution. An alpha-stable distribution shows as one possible model [traffic-stable]. This distribution can be characterized by four parameters: the localization " δ ," the stretching " γ ," the tail " α ," and the symmetry " β ," [alfa-stables]. The location parameter is highly related to the mode of the distribution: $\delta > 0$. The stretching is related to the dispersion: $\gamma > 0$. The symmetry, $-1 \leq \beta \leq 1$, indicates if the distribution is skewed to the right (the tail decays to the left) for positive values or the opposite direction for negatives ones. Finally, the tail α , defined in $(0,2]$, indicates if the distribution is Gaussian one when $\alpha=2$, a power-law without variance for $\alpha < 2$, and without statistic mean for $\alpha < 1$. The alpha-stable distribution is the generalization of the Central Limit Theorem for any distribution (i.e., it includes the cases without variance or mean).

Then, the $\phi(t)$ estimation involves the subtraction of two alpha-stable random variables, which yields on another alfa-stable distribution but symmetrical [alfa-stables]. Due to this result's characteristic, i.e., a fixed mode and symmetry, a good estimator of the mode is the median.

Therefore, sic performs periodic measurements to infer the difference of two clocks on the Internet, taking advantage of the empiric observations. The periodicity of RTT measurements is set to 1 second.

The parameters of the simple skew model [ToN2008] are estimated by the following equation:

$$\phi(t) = K + F * t , \quad (5)$$

where $\phi(t) = C_c - C_s$, K is a constant representing the absolute difference of time of client clock C_c and server clock C_s , and F is the rate parameter. As sic frequency is a differential clock, we only estimate the frequency parameter " F ."

Note that the "K" parameter cannot be estimated using just endpoints measurements. Assessing the "K" parameter accurately is out of scope, and we use $K = \min(RTT)/2$, as it is used in several synchronization protocols under the assumption of symmetric paths, e.g, NTP. Considering the following asymmetry definition,

$$A = 1 - \frac{t[c \rightarrow s]}{t[c \leftarrow s]}, \quad (6)$$

where $t[c \rightarrow s]$ is the minimum delay measured from the client to the server. The maximum asymmetry A of equation 6 is $A=1$, which is unlucky, and this establishes the hardbound for the error of K as $\min(RTT)$: if $t[c \rightarrow s]$ approaches RTT , $t[c \rightarrow s]$ approaches zero. The difference between the two is $\phi(t)$, and this difference hence is close to $\min(RTT)$, if $A=1$. In our experiments (see Section 4.1), the error in estimation $\phi(t)$ was always less than $\min(RTT)/2$.

Another problem with most of the synchronization protocols is the minimum RTT estimation, which depends upon the time-window within which the RTT is captured. A minimum RTT can only be measured in the absence of any cross traffic. In the first step, the minimum RTT measured during a window of 10 minutes ($mRTT_{10m}$) is captured. Based on these values, the minimum RTT over a week ($mRTT_w$) is determined. RTT_{ee} is defined as $mRTT_{10m} - mRTT_w$. Figure 3 shows the the RTT estimation error captured during an experiment where the minimum the latency between probes was 9431 microseconds during one week, i.e., $mRTT_w=9431$ microseconds. Notice that $mRTT_{10m}$ varies a lot, and the observed values can be more than 450 microseconds above the minimum RTT over a week. This error is a consequence of the statistical behavior of the RTT, which can be modeled by the alfa-stable distribution.

Finally, it is mostly believed there always exist NTP servers at less than five hops with few milliseconds of RTT because of the NTP deployment. In Appendix A we show a typical case in Latin America region where the RTT differs notably from a host in the same city (Buenos Aires). This example reveals that some countries could not have this desired situation, and other synchronization tools are needed.

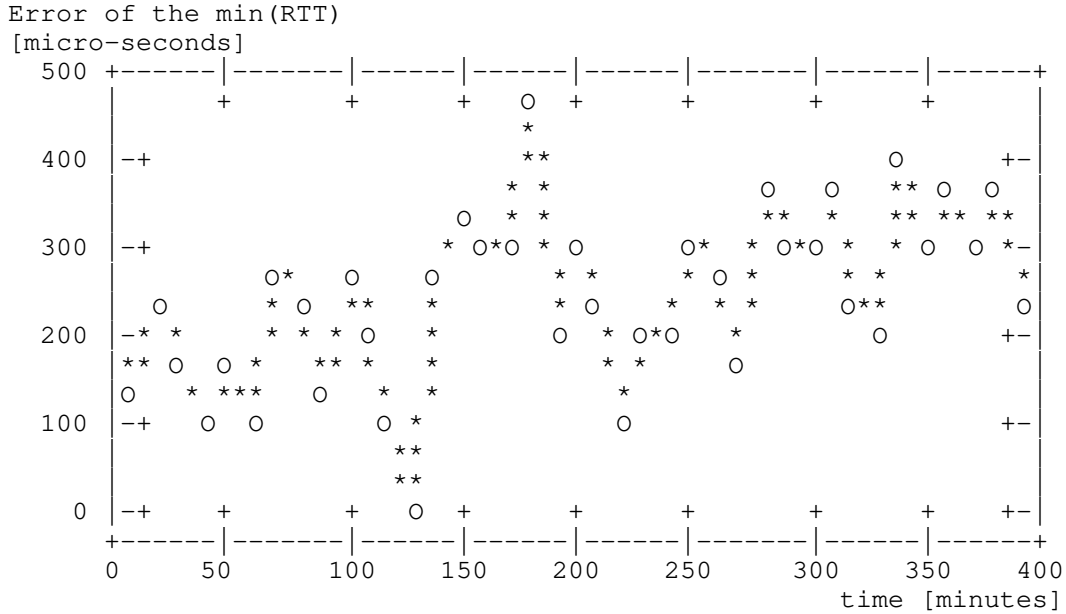


Figure 3: Min RTT error, estimated every 10 minutes along 7 hours.--

The sic frequency protocol estimates $\phi(t)$ of Equation 5 using measurement statistics and taking advantage of the inherent RTT properties, i.e., the heavy tail distribution and its alpha-stable distribution model. The basic sic frequency operation is to periodically send packets, estimate $\phi(t)$, and correct the local clock with:

$$t_c = t + \phi(t) , \quad (7)$$

where t_c is the corrected time and t the local clock time (notice that $\phi(t)$ is calculated according to Equation 1).

The sic protocol also detects route changes by seeking a non-negligible difference between the minimum RTT of the actual and past round trip measurement. The next section also discusses different mechanisms to detect route changes by RTT evaluation.

3. The formal definition of sic frequency protocol

Section 3.1 presents the sic frequency algorithm. In addition, parameters and their definitions are introduced. Finally, formal packet formats are provided.

The sic frequency protocol MUST sign the packets with the deterministic Elliptic Curve Digital Signature Algorithm (ECDSA) specified by [RFC6979] to protect sic frequency from MitM attacks. To avoid delays when a packet is signed, sic frequency signs them in a deferred fashion. That is, in each packet carries the signature of the previous packet (see algorithms in Figure 6 and Figure 5).

3.1. Algorithm description

A sic frequency implementations MUST support the formal description specified by this section. Once activated, the sic frequency protocol MUST operate permanently while a client and a receiver exchange measurement packets. The sic frequency works with three states: NOSYNC, PRESYNC and SYNC. These states are triggered by the variables errsync, presync, and sync.

Lines 1 to 4 of the pseudocode in Figure 4 initialize the required data structures needed and set the sic frequency state to NOSYNC. In NOSYNC state, a complete measurement window estimates ϕ 's by Equation 2 (see line 8). Notice that also Equation 3 can be used, or an average of both Equations. During the experiments, using a single equation only resulted in estimations with a smaller error. The possible explanation is that measurements are affected by the same type of traffic.

The median of the measurement window is computed in line 9, while lines 10-12 are used to verify if the path changed during the measurements. When an appreciable difference is detected (bounded by errRTT) in line 13, the "else" clause is executed and the systems re-initiates the cycle (see lines 17-22). Notice that line 13 verifies if the minimum RTTs' absolute value is lower than a percentage of minimum over the complete RTT window.

The sic three tables of pseudocode present frequency algorithm specification. The parameters are explained after the third table.

```

=====
|                               sic frequency algorithm                               |
=====
1  Wmedian <-0, Wm <-0, WRTT <-0, actual_m <-0, actual_c <-0
2  presync <- INT_MAX - P, epochsync <- INT_MAX - P, n_to <-0
3  synck <- false, errsync <- epoch, set(0, 0, NOSYNC), e_prev<-epoch
4  send_sic_packet(SERVER_IP, TIMEOUT)
5  for each timer(RUNNING_TIME) == 0
6      (epoch, t1, t2, t3, t4, to) <- send_sic_p(SERVER_IP, TIMEOUT)
7      if (to == false) then
8          Wm <- t1 - t2 + (t2 - t1 + t4 - t3)/2
9          Wmedian <- median(Wm)
10         WRTT <- t4 - t1 size(W)
11         RTTf <- min(WRTT[size(WRTT)/2,size(WRTT)])
12         RTTl <- min(WRTT[0,size(WRTT)/2])
13         if ((|RTTf - RTTl| <= errRTT * min(WRTT)) then
14             if (epoch >= presynck + P)) then
15                 presynck <- true
16             end if
17         else
18             synck <- false, Wmedian <- 0
19             Wm <- 0, errsync <- epoch, n_to <- 0
20             epoch_sync <- INT_MAX - P, pre_sync <- INT_MAX - P
21             set(0, 0, NOSYNC)
22         end if
23         if ((synck == true) && (epoch >= epochsync + P)) then
24             (m, c) <- linear_fit(Wmedian)
25             actual_c <- c
26             actual_m <- (1-alpha) * m + alpha * actual_m
27             epochsync <- epoch, n_to <- 0
28             set(actual_m, actual_c, SYNC)
29         else
30             if (epoch == errsync + MEDIAN_MAX_SIZE) then
31                 presync <- epoch
32             end if
33             if (epoch >= presync + P) then
34                 (actual_m, actual_c) <- linear_fit(Wmedian)
35                 synck <- true , epoch_sync <- epoch
36                 set(actual_m, actual_c, PRESYNC)
37             end if
38         end if
39     else
40         to <- false
41     end if
42 end for
=====

```

Figure 4: Formal description of sic.--

Several conditions should be verified to pass from NOSYNC to PRESYNC. First, the "else" condition of line 29 should occur, and also the elapsed time between errsync and actual epoch should be MEDIAN_MAX_SIZE (30-32). Therefore, when it also P time is passed from presync, the condition on line 33 is true, and the system arrives at PRESYNC, providing an initial estimation of phi.

Then, if there is no route change, the condition in line 14 will be true when the time was increased in another P period. Then, the system is in the SYNC state, and it provides the estimation of $\phi(t)$ in line 28. Notice that every P time, the estimation of $\phi(t)$ is computed unless a route change occurs (lines 13 and 17-22).

The function in line 6: (epoch, t1, t2, t3, t4, to) <- send_sic_packet(SERVER_IP, TIMEOUT), has a special treatment. It sends the packets specified in Section 3.3, which have signatures. To avoid the processing delay caused by the signature computation, we implemented a policy to send the signature of the previous packet, and if an error is detected, we can stop the synchronization just one loop ahead.

Figure 5 illustrates how the client-side MUST implement the function send_sic_p (SERVER_IP, TIMEOUT). This function computes the timestamp t1 in line 1 and builds and sends the UDP packet in lines 2-3. Then, if there is no timeout, it calculates the t4 timestamp (line 5), and if no packets were lost, verifies the signature of the previous one in lines 8-18. If the signature is not valid with the received certificate, then the system MUST change to NOSYNC state immediately (see line 11). NOSYNC state MUST also be set if the limit of time without receiving packets MAX_to is reached. Finally, it stores the received packet into prev_rcv_pck (a global variable) to use in the next packet (line 19). Notice that n_to, the lost packets, is a global variable, as well as the epoch of the previous packet: e_prev.

```

=====
|                               function: send_sic_p(server, TIMEOUT)                               |
=====
1  t1 <- get_timestamp()
2  sic_P <- sic_pck(t1, 0, 0, prev_sig)
3  (to, rcv_sic_pck) <- send(sic_P,UDP_PORT, SERVER_IP, TIMEOUT)
4  if (to == false) then
5      t4 <- get_timestamp()
6      epoch <- trunc_to_seconds(t1)
7      prev_sig <- get_signature(sic_P)
8      if (epoch - e_prev <= RUNNING_TIME) then
9          if (n_to < MAX_to) then
10             if (verify(prev_rcv_pck,rcv_sic.CERT) == false) then
11                 set(0, 0, NOSYNC)
12             else
13                 n_to <- 0,  e_prev <- epoch
14             end if
15         else
16             set(0, 0, NOSYNC)
17         end if
18     end if
19     prev_rcv_pck <- rcv_sic_pck
20     t2 <- rcv_sic_pck.t2
21     t3 <- rcv_sic_pck.t3
22 else
23     n_to <- n_to + 1
24 end if
25 return (epoch, t1, t2, t3, t4, to)
=====

```

Figure 5: The send_sic_p function.--

The server sic algorithm is presented in Figure 6. It uses `prev_sic_P{}`, which is a structure to store the received previous signatures, indexed by the IP client addresses (`CLIENT_add` contains its IP and UDP port), and the same for `prev_sig{}` with the previously sent signatures. Line 6 verifies either signature is null because it is the first packet or a valid signature. In both cases, the algorithm process the packet computing `t3`, building up the sic frequency packet, sending it, and computing its signature (stored to send in the next reply) in lines 7-11. Next, the actual packet is stored in the `prev_sic_P{}` structure, line 13.

```

=====
|                               sic Server algorithm                               |
=====
1  prev_sic_P{} <- null, prev_sig{} <-- null
2  while (RUNNING == true) then
3      if (receive() == true) then
4          t2 <- get_timestamp()
5          prev_sig <- get_signature(prev_sic_P{receive().CLIENT_add})
6          if (prev_sig == null) ||
              (verify(prev_sig, CLIENT_add.CERT) == true) then
7              t3 <- get_timestamp()
8              sic_P <- sic_pack(t1, t2, t3, prev_sig)
9              send(sic_P, CLIENT_add.UDP, CLIENT_add.IP, TIMEOUT)
10             prev_sig <- get_signature(sic_P)
11             prev_sig{receive().CLIENT_add} <- prev_sig
12         end if
13     prev_sic_P{receive().CLIENT_add} <- receive().sic_pack
14 end if
15 end while
=====

```

Figure 6: Algorithm sic for the Server.--

3.2. Protocol definitions

We provide a formal definition of each used constant and variables; the RECOMMENDED values are displayed in parentheses at the end of the description. These constant and variables MUST be represented in a sic frequency implementation. All the types MUST be respected. They are expressed in "C" programming language running on a 64-bit processor.

a. Constants used for the sic frequency algorithm (Figure 4)

1. RUNNING_TIME: is the period between sic packets are sent (1 second).
2. MEDIAN_MAX_SIZE: is the window size used to compute the median of the measurements (600).
3. P: is the period between phi's estimation (60).
4. alpha: is a float in the [0,1], the coefficient of the autoregressive estimation of the slope of phi(t) (0.05).
5. TIMEOUT: is the maximum time in seconds that a sic packet reply is expected (0.8 seconds).

6. SERVER_IP: is the IP address of the server (@IP in version 4 or 6).
 7. errRTT: is a float that bounds the maximum difference to detect a route change (0.2).
 8. MAX_to: is an integer representing the maximum number of packet lost ($P/10$).
 9. CERT: is a public certificate of the other end, it is used to verify signs of the packets.
 10. UDP_PORT: is an integer with the port UDP where the service is running on the server. (4444)
 11. SERVER_IP: is the IP address of the server.
 12. CLIENT_IP: is the IP address of the client.
- b. States used for the sic frequency algorithm (Figure 4)
1. NOSYNC: a boolean indicates that it is not possible to correct the local time.
 2. PRESYNC: an integer indicates that sic is almost (P RUNNING_TIME) seconds from the synchronization.
 3. SYNC: a boolean indicates that sic is synchronized.
- c. Variables used for the sic frequency algorithms (Figure 4, Figure 5 and Figure 6)
1. errsync: is an integer with the UNIX timestamp epoch of the initial NOSYNC cycle. It is used to complete the window or measurements (W_m) to compute their medians.
 2. presync: is an integer with the UNIX timestamp epoch of the initial PRESYNC cycle. It is used to wait until (P RUNNING_TIME) seconds to the linear fit of $\phi(t)$.
 3. synck: is an integer with the UNIX timestamp epoch of the initial SYNC cycle. Every P RUNNING_TIME) seconds the $\phi(t)$ function is estimated.
 4. epochsync: is an integer with the last UNIX timestamp epoch of synchronization. It is used to compute a new estimation of $\phi(t)$, every (P RUNNING_TIME) seconds.

5. epoch: is an integer with UNIX timestamp in seconds. It carries the initial epoch of each sic measurement packet.
6. t1, t2, t3, t4: are long long integers to store the t UNIX timestamps in microseconds.
7. actual_m : is a double with the slope for the $\phi(t)$ estimation.
8. actual_c: is a double with the intercept for the $\phi(t)$ estimation.
9. Wm: is an array of doubles of MEDIAN_MAX_SIZE. It stores the instantaneous estimates of $\phi(t)$.
10. Wmedian: is an array of doubles of P size. It saves the computed medians of Wm every RUNNING_TIME.
11. WRTT: is an array of doubles of (2 P) size. It stores the calculated RTT of last measurements.
12. RTTl: is a double with the minimum of last P RTTs. It is used to detect changes on the route from the client to the server.
13. RTTf: is a double with the minimum of previous P RTTs. It is used to detect changes on the route from the client to the server.
14. n_to: is an integer representing the number of lost packets in the actual synchronization window P.
15. e_prev: is an integer with the UNIX timestamp epoch of the last valid packet.
16. prev_rcv_pck: is a sic packet structure, the previous received one.

3.3. Protocol packet specification

The sic frequency uses UNIX microsecond format timestamps. Regarding Figure 2, the client takes a timestamp t1 just before it sends the packet. When the server receives the packet, it immediately computes t2, and just before it is sent back to the client, it computes t3. When the client receives the packet, it calculates t4.

The server does not need the timestamp t_1 because the proposed protocol synchronizes a client with the server clock. This information could, however, be useful for the server for future use.

The packets respect the NTP4 format as they are defined in the Section A.1.2 of [RFC5905] and the signature of the fields, shown in Figure 7. We use the time formats of 64 bits with seconds and a fraction of seconds. They MUST be sent as UDP data, and it MUST have the mentioned fields.

The client and server certificates SHOULD be valid and signed ones (only for experimentation, the user MAY use autogenerated ones).

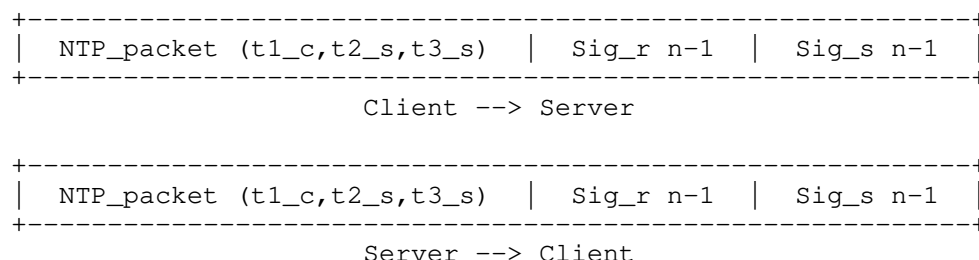


Figure 7: Packet format for the sic protocol.--

3.4. Minimum sic deployment

To deploy the sic frequency algorithm, as a minimum, a Server and one Client are needed. The Server can support multiple clients. The maximum number of clients is for further study. The Server clock is considered the master one, and all clients synchronize with it. The Server-side runs sic frequency as a server with a UDP_PORT number, as specified by the algorithm shown in Figure 6.

Client sic runs the algorithm shown in Figure 4 and also SHOULD provide the corrected time as

$$t = \text{actual_c} + \text{actual_m} * \text{timestamp} \quad (8)$$

Figure 8

Different ways of doing this task are possible:

Providing a client capable of reading the variables `actual_m` and `actual_c` in shared memory and producing the result of Equation 8.

Providing a service in a UDP port answering the correct timestamp queries with Equation 8.

Other solution.

4. Implementation of sic frequency protocol

In this section we present the prove of the sic concept through some test that we already performed, and the current implementation of sic in C language. Our implementation is publicly available [[sic-implementation](#)].

This protocol implements protection against MitM attacks. The identity of endpoints is guaranteed by signed certificates using the deterministic Elliptic Curve Digital Signature Algorithm (ECDSA) specified in the [RFC6979]. Server and Client should use signed and valid ECDSA certificates to ensure their identity, and each side has responsible to verify the public certificate of the other side before to run the algorithm in Figure 4.

4.1. Evaluation

To verify the sic proposal, we tested it using three hosts with GPS units. The first two were located in Buenos Aires, and the third at Los Angeles. We slightly modified the algorithm in Figure 4 to trigger each measurement using the PPS (pulse per second) the signal provided by the GPS unit. Then, recording the client and server clocks with the PPS signal, we can determine the real ϕ function of Equation 1, within the GPS error (it is several orders of magnitude smaller than the error of the sic frequency protocol).

We use MTIE defined as follows (Maximum Time Interval Error, see [ToIM1996]):

$$\text{MTIE} = \max [\phi(t')] - \min [\phi(t)] , \quad (9)$$

for every t' and t in the interval $[t, t+s]$; and we chose $s=60$ seconds. We first used two host (RaspBerriesPI-2) connected back to back to analyze the minimum achievable precision, yielding an MTIE of 15.8 microseconds for the 90 percentile. Then, we selected two real cases of study, one national and the other international. In Figure 9 we show the result of the MTIE, evaluated in 60 seconds intervals, for the experiment Buenos Aires-Buenos Aires (RTT of 10ms) and Buenos Aires-Los Angeles (RTT of 198ms). The percentile 90

corresponds to 18.35 microseconds for the Buenos Aires case, and 25.4 microseconds for the Los Angeles case. The percentile 97.5 corresponds to 30 microseconds for the Buenos Aires case, and 42 microseconds for the Los Angeles case. We display the quartiles in Figure 10. These measurements were performed during a week in each case.

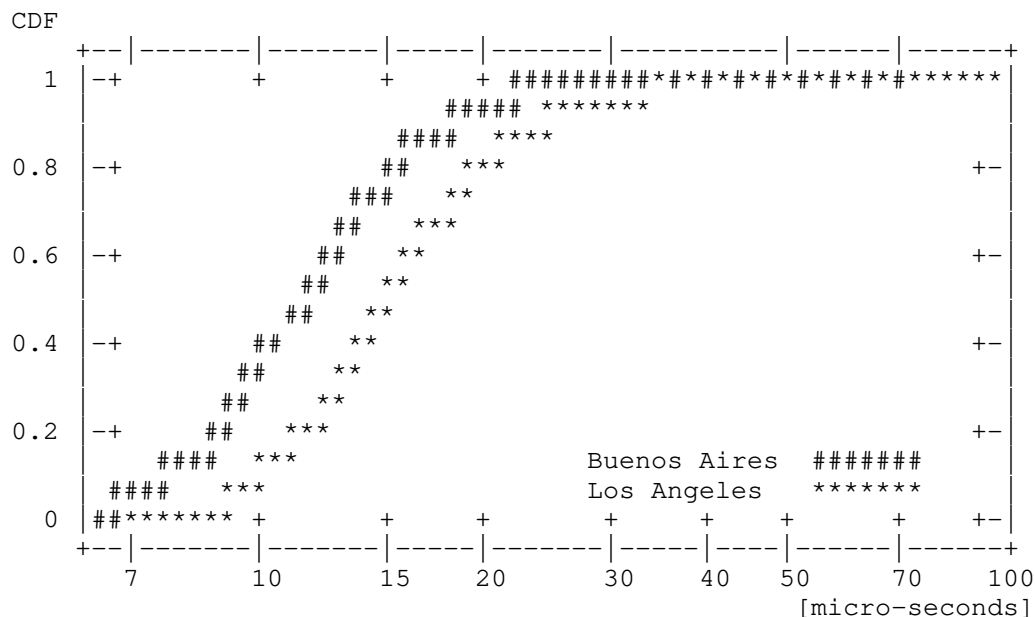


Figure 9: Cumulative distribution function of the MTIE (60s).--

	Buenos Aires (10ms)	Los Angeles (198ms)	local NTP4 (12ms)
Q3	14.69	19.29	9059
Q2	11.60	14.93	5245
Q1	9.41	12.26	3338

Figure 10: Table with MTIE quartiles for two RTT sic cases, and for a local NTP4 system (the numbers indicate microseconds).--

We also conducted another test for NTP4 in Buenos Aires, Argentina. We used a host with GPS, whose PPS signal triggered a process to log actual timestamps. This host was also running NTP4 with the server time.afip.gov.ar, also located in Buenos Aires city, with an average

RTT of 12ms. Applying the same process of the previous cases, we obtained that the following quartiles Q3: 9.1ms, Q2: 5.2ms, and Q1: 3.3ms for the MTIE of the NTP4 measurements (also reported in Figure 10). Finally, percentile 90 of the NTP4's MTIE is 11.1ms.

The comparison of NTP4 with frequency sic shows that this new method performs two orders of magnitude better.

5. Conclusions

This document presents the sic frequency protocol, employed to synchronize host clock frequency across the Internet, and which is also resistant to MitM attacks. The main field of application is the Internet performance analysis, e.g., to measure congestion, jitter, or delay variations parameters; all of them are based on a difference of timestamps. It shows the complete specification, implementation, and experiment results that support its working principle. In particular, sic frequency provides a clock rate stability of less than 1ppm for most of the time.

6. Security Considerations

Following [RFC7384] enumeration of Time Protocols in packet-switched networks, the proposed encryption of timing packets, based on a mechanism of secure key distribution, provides the following characteristics:

3.2.1 Packet Manipulation: Prevented by packet signature.

3.2.2 Spoofing: Prevented by packet signature and secure key distribution.

3.2.3 Replay Attack: Prevented by chain signing of packets.

3.2.4 Rogue Master Attack: Prevented by secure key distribution.

3.2.5 Packet Interception and Removal: If several packets are removal, the protocol does not arrive at SYNC state.

3.2.6 Packet Delay Manipulation: Not prevented. Future versions may prevent this using over-specification of timing (using redundant masters)

3.2.7 L2/L3 DoS attacks: Not prevented. This attack can be prevented in future versions using over-specification of timing and redundant masters time servers.

3.2.8 Cryptographic performance attacks: Not an issue in ECDSA.

3.2.9 DoS attacks against the time protocol: Prevented by secure key distribution.

3.2.10 Grandmaster Time source attack (GPS attacks): Not prevented. Future versions may prevent this using over-specification of timing (using several time servers) .

3.2.11 Exploiting vulnerabilities in the time protocol: Not prevented, future vulnerabilities are unknown.

3.2.12 Network Reconnaissance: Not prevented in this version. No countermeasures were done in node anonymization.

The Packet Delay manipulation is one of the hardest problems to solve because there exist some smart ways to attack any synchronization protocol. Even though, the sic frequency protocol can protect itself because it can identify several attacks of this type, i.e., it is challenging to mimic traffic behavior. This emulation of behavior can be easily overcome regarding the RTTs' distribution, which has to be a heavy-tailed one. This way, the analysis should be studied to ensure the implementation of a robust and light test of the raw data.

7. IANA Considerations

This memo makes no requests of IANA.

8. Acknowledgements

The authors thank to Ethan Katz-Bassett, Zahaib Akhtar, the USC and CAIDA for lodging the testbed of the sic frequency protocol.

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>>.

- [RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <<https://www.rfc-editor.org/info/rfc6979>>.
- [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>>.
- [RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <<https://www.rfc-editor.org/info/rfc7679>>.
- [RFC8173] Shankarkumar, V., Montini, L., Frost, T., and G. Dowd, "Precision Time Protocol Version 2 (PTPv2) Management Information Base", RFC 8173, DOI 10.17487/RFC8173, June 2017, <<https://www.rfc-editor.org/info/rfc8173>>.

9.2. Informative References

- [alfa-stables]
Uchaikin, V.V. and V.M. Zolotarev, "Chance and stability: stable distributions and their applications.", Walter de Gruyter. (Book), 1999.
- [ITU-G.8260]
ITU, T. S. S. O., "Definitions and terminology for synchronization in packet networks (Recommendation ITU-T G.8260)", August 2015.
- [sic-implementation]
Samariego, E., Ortega, A., and J.I. Alvarez-Hamelin, "Synchronizing Internet Clocks", <https://github.com/CoNexDat/SIC>, February 2018.
- [ToIM1996] Bregni, S., "Measurement of maximum time interval error for telecommunications clock stability characterization", Bregni S. Measurement of maximum time interval error for telecommunicIEEE transactions on instrumentation and measurement. 45(5):900-906, October 1996.
- [ToN2008] Veitch, D., Ridoux, J., and S.D. Korada, "Robust synchronization of absolute and differential clocks over networks.", IEEE.ACM Transactions on Networking (TON) 17.2 (2009): 417-430, 2009.

[traffic-stable]

Carisimo, E., Grynberg, S.P., and J.I. Alvarez-Hamelin,
 "Influence of traffic in stochastic behavior of
 latency.", 7th PhD School on Traffic Monitoring and
 Analysis (TMA), Ireland, Dublin,, June 2017.

Appendix A. Example of RTT to NTP servers

This appendix shows an experiment to measure the RTT and the distance in hops from four different points to a time server in Buenos Aires city (the capital of Argentina). We did the measures two times from the four points, and we used one hundred packets to determine some statistical parameters. Next traceroute measurements show that the number of hops and RTT are very different from each point also changes a lot. For instance, taking a distinctive look at the STD, average, and maximum is possible to detect huge variations. We provide here a case in Argentina, trying to reach an NTP server from 4 different points at the Buenos Aires city.

```
-----
host1$ mtr -r -c 100 time.afip.gov.ar
Start: Tue Mar 27 19:03:51 2018
HOST: raspbian-server
  1. | -- gw-vlan-srv.innova-red.ne 0.0% 100 2.2 2.8 2.1 37.7 4.9
  2. | -- rnoc5.BUENOS-AIRES.innova 0.0% 100 2.3 3.8 2.1 55.8 7.9
  3. | -- 10.5.10.2 0.0% 100 2.5 2.6 2.2 3.1 0.0
  4. | -- 200.0.17.104 0.0% 100 3.1 3.1 2.4 13.7 1.1
  5. | -- 172.18.2.53 0.0% 100 4.8 5.7 3.8 12.4 1.7
  6. | -- time.afip.gob.ar 0.0% 100 5.2 5.2 3.8 12.0 1.3
```

```
host1$ mtr -r -c 100 time.afip.gov.ar
Start: Tue Mar 27 18:57:06 2018
HOST: raspbian-server
  1. | -- gw-vlan-srv.innova-red.ne 0.0% 50 2.4 2.8 2.0 34.2 4.5
  2. | -- rnoc5.BUENOS-AIRES.innova 0.0% 50 2.1 3.8 2.1 52.8 7.7
  3. | -- 10.5.10.2 0.0% 50 2.7 2.9 2.2 15.6 1.8
  4. | -- 200.0.17.104 0.0% 50 2.8 3.0 2.3 3.9 0.0
  5. | -- 172.18.2.53 0.0% 50 4.5 5.8 3.8 17.8 2.2
  6. | -- time.afip.gob.ar 0.0% 50 4.7 9.9 4.2 238.5 33.0
```

```
-----
host2$ mtr -r -c 100 time.afip.gov.ar
Start: Tue Mar 27 19:03:47 2018
HOST: ws-david
  1. | -- 10.10.96.1 0.0% 100 0.3 0.2 0.2 0.3 0.0
  2. | -- 200.16.116.171 0.0% 100 1.0 5.9 0.6 158.4 22.9
```

3.	-- static.33.229.104.190.cps	1.0%	100	1.6	2.5	1.5	80.6	8.0
4.	-- static.129.192.104.190.cp	0.0%	100	2.1	1.9	1.8	3.0	0.1
5.	-- 200.0.17.104	1.0%	100	2.0	2.2	1.8	9.4	0.7
6.	-- 172.18.2.53	0.0%	100	3.2	4.2	3.1	12.0	1.5
7.	-- auth.afip.gob.ar	0.0%	100	4.2	4.5	3.3	9.8	1.2

```
host2$ mtr -r -c 100 time.afip.gov.ar
```

```
Start: Tue Mar 27 18:57:00 2018
```

HOST:	ws-david	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1.	-- 10.10.96.1	0.0%	50	0.3	0.3	0.2	0.7	0.0
2.	-- 200.16.116.171	0.0%	50	0.9	6.7	0.7	196.5	29.1
3.	-- static.33.229.104.190.cps	2.0%	50	1.6	1.7	1.5	2.2	0.0
4.	-- static.129.192.104.190.cp	0.0%	50	2.1	2.0	1.7	2.4	0.0
5.	-- 200.0.17.104	0.0%	50	2.0	2.1	1.8	2.6	0.0
6.	-- 172.18.2.53	0.0%	50	4.8	4.3	3.2	9.1	1.3
7.	-- time.afip.gob.ar	0.0%	50	4.3	9.4	3.3	234.9	32.7

```
host3$ mtr -r -c 100 time.afip.gov.ar
```

```
Start: 2018-03-27T19:03:51-0300
```

HOST:	aleph.local	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1.	-- 10.17.71.254	0.0%	100	4.7	30.3	3.5	280.4	52.4
2.	-- 10.255.254.250	0.0%	100	2.5	31.1	1.8	285.4	59.2
3.	-- 209.13.133.10	0.0%	100	30.5	43.9	2.3	237.2	64.9
4.	-- host169.advance.com.ar	3.0%	100	36.0	64.8	3.1	404.4	86.9
5.	-- 200.32.33.33	2.0%	100	106.9	70.6	2.8	315.0	80.4
6.	-- 200.32.34.66	5.0%	100	93.1	56.8	2.7	336.1	74.5
7.	-- 200.32.33.38	7.0%	100	42.8	58.0	2.9	357.8	76.7
8.	-- 209.13.139.211	4.0%	100	46.2	56.7	2.8	298.8	69.9
9.	-- 209.13.139.209	1.0%	100	84.5	57.1	2.6	277.7	72.3
10.	-- 209.13.166.211	1.0%	100	43.4	63.5	3.2	390.6	78.7
11.	-- 200.32.34.137	2.0%	100	68.7	64.1	3.7	259.5	75.5
12.	-- 200.32.33.37	0.0%	100	4.1	56.9	3.3	249.6	64.3
13.	-- 200.32.34.121	3.0%	100	10.9	65.0	4.1	415.7	85.1
14.	-- 200.32.33.241	2.0%	100	252.6	61.8	3.8	355.9	74.5
15.	-- 200.16.206.198	3.0%	100	188.0	54.6	3.1	461.7	74.9
16.	-- 172.18.2.53	2.0%	100	133.4	53.1	4.3	402.1	69.2
17.	-- time.afip.gob.ar	4.0%	100	72.5	54.1	4.9	343.2	66.9

```
host3$ mtr -r -c 100 time.afip.gov.ar
```

```
Start: 2018-03-27T18:57:05-0300
```

HOST:	aleph.local	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1.	-- 10.17.71.254	0.0%	50	125.6	88.1	3.7	392.4	79.3
2.	-- 10.255.254.250	0.0%	50	62.1	54.8	2.1	333.2	68.0
3.	-- 209.13.133.10	0.0%	50	4.0	33.9	2.4	280.8	51.3
4.	-- host169.advance.com.ar	2.0%	50	4.1	21.3	2.9	236.7	40.4
5.	-- 200.32.33.33	2.0%	50	4.5	32.2	3.2	341.3	69.4

6.	-- 200.32.34.66	4.0%	50	7.7	26.0	3.5	278.8	55.8
7.	-- 200.32.33.38	2.0%	50	4.8	29.4	3.0	221.3	43.4
8.	-- 209.13.139.211	0.0%	50	84.8	40.3	3.1	250.4	53.0
9.	-- 209.13.139.209	0.0%	50	25.1	35.0	2.8	249.2	55.4
10.	-- 209.13.166.211	0.0%	50	3.7	33.5	2.6	188.9	54.3
11.	-- 200.32.34.137	0.0%	50	5.6	28.2	3.7	224.3	51.1
12.	-- 200.32.33.37	0.0%	50	3.7	24.2	3.5	189.5	44.9
13.	-- 200.32.34.121	0.0%	50	4.7	30.8	4.0	213.2	51.6
14.	-- 200.32.33.241	0.0%	50	14.4	33.1	3.9	364.6	67.2
15.	-- 200.16.206.198	0.0%	50	5.0	58.2	3.1	300.7	88.5
16.	-- 172.18.2.53	0.0%	50	9.4	117.8	4.4	315.1	103.4
17.	-- time.afip.gob.ar	0.0%	50	199.6	120.2	5.2	484.0	96.2

```

host4$ mtr -r -c 100 time.afip.gov.ar
Start: 2018-03-27T19:03:51-0300
HOST: cnet

```

		Loss%	Snt	Last	Avg	Best	Wrst	StDev
1.	-- 157.92.58.1	0.0%	100	6.6	2.8	0.3	12.8	2.5
2.	-- ???	100.0	100	0.0	0.0	0.0	0.0	0.0
3.	-- ???	100.0	100	0.0	0.0	0.0	0.0	0.0
4.	-- host98.131-100-186.static	0.0%	100	5.7	5.6	1.5	9.4	2.2
5.	-- host130.131-100-186.stati	0.0%	100	6.5	6.3	2.5	10.3	2.2
6.	-- 200.0.17.104	0.0%	100	2.4	2.7	2.3	15.6	1.4
7.	-- ???	100.0	100	0.0	0.0	0.0	0.0	0.0
8.	-- time.afip.gob.ar	0.0%	100	4.9	7.6	3.9	243.0	23.9

```

host4$ mtr -r -c 100 time.afip.gov.ar
Start: Tue Mar 27 18:41:40 2018
HOST: cnet

```

		Loss%	Snt	Last	Avg	Best	Wrst	StDev
1.	-- 157.92.58.1	0.0%	50	4.0	1.6	0.3	9.1	1.6
2.	-- ???	100.0	50	0.0	0.0	0.0	0.0	0.0
3.	-- ???	100.0	50	0.0	0.0	0.0	0.0	0.0
4.	-- host98.131-100-186.static	0.0%	50	4.7	5.5	1.5	10.9	2.4
5.	-- host130.131-100-186.stati	0.0%	50	8.4	6.5	2.6	10.5	2.2
6.	-- 200.0.17.104	0.0%	50	2.5	2.8	2.3	11.0	1.2
7.	-- ???	100.0	50	0.0	0.0	0.0	0.0	0.0
8.	-- time.afip.gob.ar	0.0%	50	4.9	9.2	3.8	226.7	31.4

Authors' Addresses

Jose Ignacio Alvarez-Hamelin (editor)
Universidad de Buenos Aires - CONICET
Av. Paseo Colon 850
C1063ACV Buenos Aires
Argentina

Phone: +54 11 5285-0716
Email: ihameli@cnet.fi.uba.ar
URI: <http://cnet.fi.uba.ar/ignacio.alvarez-hamelin/>

David Samaniego
Universidad de Buenos Aires
Av. Paseo Colon 850
C1063ACV Buenos Aires
Argentina

Phone: +54 11 5285-0716
Email: dsamanie@fi.uba.ar

Alfredo A. Ortega
Universidad de Buenos Aires
Av. Paseo Colon 850
C1063ACV Buenos Aires
Argentina

Phone: +54 11 5285-0716
Email: ortegaalfredo@gmail.com

Ruediger Geib
Deutsche Telekom
Heinrich-Hertz-Str. 3-7
64297 Darmstadt
Germany

Phone: +49 6151 5812747
Email: Ruediger.Geib@telekom.de

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

D. Franke

D. Sibold
K. Teichel
PTB
M. Dansarie

R. Sundblad
Netnod
July 02, 2018

Network Time Security for the Network Time Protocol
draft-dansarie-nts-00

Abstract

This memo specifies Network Time Security (NTS), a mechanism for using Transport Layer Security (TLS) and Authenticated Encryption with Associated Data (AEAD) to provide cryptographic security for the client-server mode of the Network Time Protocol (NTP).

NTS is structured as a suite of two loosely coupled sub-protocols: the NTS Key Establishment Protocol (NTS-KE) and the NTS Extension Fields for NTPv4. NTS-KE handles NTS service authentication, initial handshaking, and key extraction over TLS. Encryption and authentication during NTP time synchronization is performed through the NTS Extension Fields in otherwise standard NTP packets. Except for during the initial NTS-KE process, all state required by the protocol is held by the client in opaque cookies.

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Objectives	3
1.2. Protocol Overview	4
2. Requirements Language	6
3. TLS Profile for Network Time Security	6
4. The NTS Key Establishment Protocol	7
4.1. NTS-KE Record Types	9
4.1.1. End of Message	9
4.1.2. NTS Next Protocol Negotiation	10
4.1.3. Error	10
4.1.4. Warning	10
4.1.5. AEAD Algorithm Negotiation	11
4.1.6. New Cookie for NTPv4	11
4.1.7. NTP Server Negotiation	12
4.2. Key Extraction (generally)	12
4.3. Key Extraction (for NTPv4)	13
5. NTS Extension Fields for NTPv4	13
5.1. Packet Structure Overview	13
5.2. The Unique Identifier Extension Field	14
5.3. The NTS Cookie Extension Field	14
5.4. The NTS Cookie Placeholder Extension Field	14
5.5. The NTS Authenticator and Encrypted Extension Fields Extension Field	15
6. Protocol Details	17
7. Suggested Format for NTS Cookies	21
8. Usage of NTP pools	22
9. IANA Considerations	23
9.1. Service Name and Transport Protocol Port Number Registry	23
9.2. TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs Registry	23
9.3. TLS Exporter Labels Registry	24

9.4.	NTP Kiss-o'-Death Codes Registry	24
9.5.	NTP Extension Field Types Registry	24
9.6.	Network Time Security Key Establishment Record Types Registry	25
9.7.	Network Time Security Next Protocols Registry	26
9.8.	Network Time Security Error and Warning Codes Registries	27
10.	Security Considerations	28
10.1.	Sensitivity to DDoS attacks	28
10.2.	Avoiding DDoS Amplification	28
10.3.	Initial Verification of Server Certificates	29
10.4.	Delay Attacks	30
10.5.	Random Number Generation	30
11.	Privacy Considerations	30
11.1.	Unlinkability	31
11.2.	Confidentiality	31
12.	Acknowledgements	32
13.	References	32
13.1.	Normative References	32
13.2.	Informative References	33
Appendix A.	Terms and Abbreviations	34
Authors' Addresses	35

1. Introduction

This memo specifies Network Time Security (NTS), a cryptographic security mechanism for network time synchronization. A complete specification is provided for application of NTS to the client-server mode of the Network Time Protocol (NTP) [RFC5905].

1.1. Objectives

The objectives of NTS are as follows:

- o Identity: Through the use of the X.509 public key infrastructure, implementations may cryptographically establish the identity of the parties they are communicating with.
- o Authentication: Implementations may cryptographically verify that any time synchronization packets are authentic, i.e., that they were produced by an identified party and have not been modified in transit.
- o Confidentiality: Although basic time synchronization data is considered non-confidential and sent in the clear, NTS includes support for encrypting NTP extension fields.

- o Replay prevention: Client implementations may detect when a received time synchronization packet is a replay of a previous packet.
- o Request-response consistency: Client implementations may verify that a time synchronization packet received from a server was sent in response to a particular request from the client.
- o Unlinkability: For mobile clients, NTS will not leak any information additional to NTP which would permit a passive adversary to determine that two packets sent over different networks came from the same client.
- o Non-amplification: Implementations (especially server implementations) may avoid acting as distributed denial-of-service (DDoS) amplifiers by never responding to a request with a packet larger than the request packet.
- o Scalability: Server implementations may serve large numbers of clients without having to retain any client-specific state.
- o Resilience: Attacks on or faults in parts of the NTS infrastructure should not completely prohibit clients from performing time synchronization.

1.2. Protocol Overview

The Network Time Protocol includes many different operating modes to support various network topologies. In addition to its best-known and most-widely-used client-server mode, it also includes modes for synchronization between symmetric peers, a control mode for server monitoring and administration, and a broadcast mode. These various modes have differing and partly contradictory requirements for security and performance. Symmetric and control modes demand mutual authentication and mutual replay protection. Additionally, for certain message types control mode may require confidentiality as well as authentication. Client-server mode places more stringent requirements on resource utilization than other modes, because servers may have vast number of clients and be unable to afford to maintain per-client state. However, client-server mode also has more relaxed security needs, because only the client requires replay protection: it is harmless for stateless servers to process replayed packets. The security demands of symmetric and control modes, on the other hand, are in conflict with the resource-utilization demands of client-server mode: any scheme which provides replay protection inherently involves maintaining some state to keep track of what messages have already been seen.

This memo specifies NTS exclusively for the client-server mode of NTP. To this end, NTS is structured as a suite of two protocols:

The "NTS Extension Fields for NTPv4" are a collection of NTP extension fields for cryptographically securing NTPv4 using previously-established key material. They are suitable for securing client-server mode because the server can implement them without retaining per-client state. All state is kept by the client and provided to the server in the form of an encrypted cookie supplied with each request. On the other hand, the NTS Extension Fields are suitable **only** for client-server mode because only the client, and not the server, is protected from replay.

The "NTS Key Establishment" protocol (NTS-KE) is a mechanism for establishing key material for use with the NTS Extension Fields for NTPv4. It uses TLS to exchange keys, provide the client with an initial supply of cookies, and negotiate some additional protocol options. After this exchange, the TLS channel is closed with no per-client state remaining on the server side.

The typical protocol flow is as follows: The client connects to an NTS-KE server on the NTS TCP port and the two parties perform a TLS handshake. Via the TLS channel, the parties negotiate some additional protocol parameters and the server sends the client a supply of cookies along with a list of one or more IP addresses to NTP servers for which the cookies are valid. The parties use TLS key export [RFC5705] to extract key material which will be used in the next phase of the protocol. This negotiation takes only a single round trip, after which the server closes the connection and discards all associated state. At this point the NTS-KE phase of the protocol is complete. Ideally, the client never needs to connect to the NTS-KE server again.

Time synchronization proceeds with one of the indicated NTP servers over the NTP UDP port. The client sends the server an NTP client packet which includes several extension fields. Included among these fields are a cookie (previously provided by the key exchange server) and an authentication tag, computed using key material extracted from the NTS-KE handshake. The NTP server uses the cookie to recover this key material and send back an authenticated response. The response includes a fresh, encrypted cookie which the client then sends back in the clear in a subsequent request. (This constant refreshing of cookies is necessary in order to achieve NTS's unlinkability goal.)

Figure 1 provides an overview of the high-level interaction between the client, the NTS-KE server, and the NTP server. Note that the cookies' data format and the exchange of secrets between NTS-KE and

NTP servers are not part of this specification and are implementation dependent. However, a suggested format for NTS cookies is provided in Section 7.

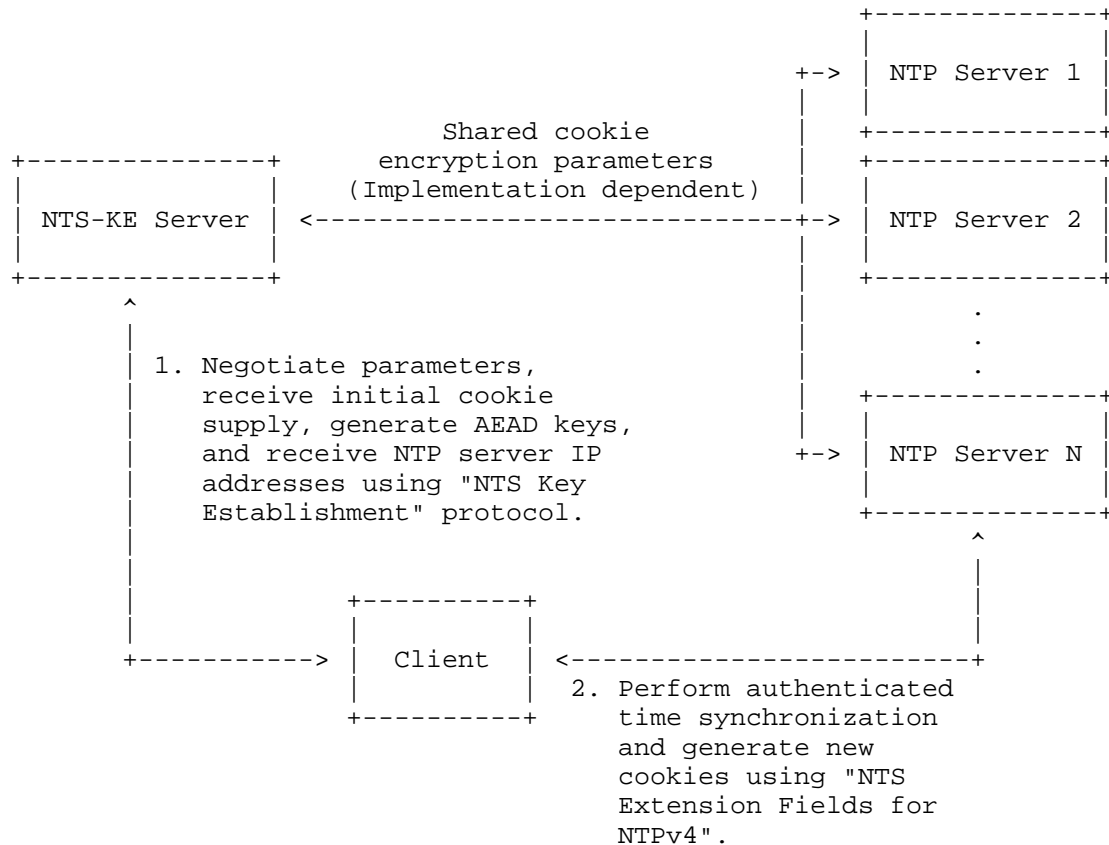


Figure 1: Overview of High-Level Interactions in NTS

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. TLS Profile for Network Time Security

Network Time Security makes use of TLS [RFC8446] for NTS key establishment.

Since securing time protocols is (as of 2018) a novel application of TLS, no backward-compatibility concerns exist to justify using obsolete, insecure, or otherwise broken TLS features or versions. We therefore put forward the following requirements and guidelines, roughly representing 2018's best practices:

Implementations MUST NOT negotiate TLS versions earlier than 1.3.

Implementations willing to negotiate more than one possible version of TLS SHOULD NOT respond to handshake failures by retrying with a downgraded protocol version. If they do, they MUST implement TLS Fallback SCSV [RFC7507].

Use of the Application-Layer Protocol Negotiation Extension [RFC7301] is integral to NTS and support for it is REQUIRED for interoperability.

4. The NTS Key Establishment Protocol

The NTS key establishment protocol is conducted via TCP port [[TBD1]]. The two endpoints carry out a TLS handshake in conformance with Section 3, with the client offering (via an ALPN [RFC7301] extension), 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 followed by a TLS "Close notify" alert and then discard the channel state.

The client's request and the server's response each SHALL consist of a sequence of records formatted according to Figure 2. Requests and non-error responses 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.

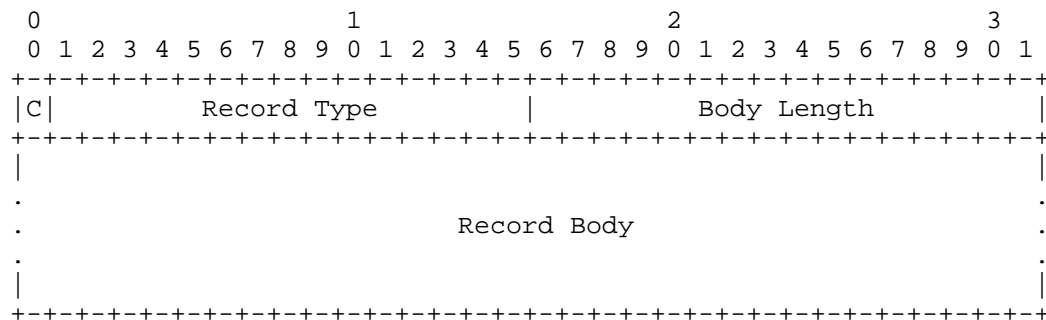


Figure 2: NTS-KE Record Format

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.

Record Type Number: A 15-bit integer in network byte order. The semantics of record types 0-6 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 C, 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 3 provides a schematic overview of the key exchange. It displays the protocol steps to be performed by the NTS client and server and record types to be exchanged.

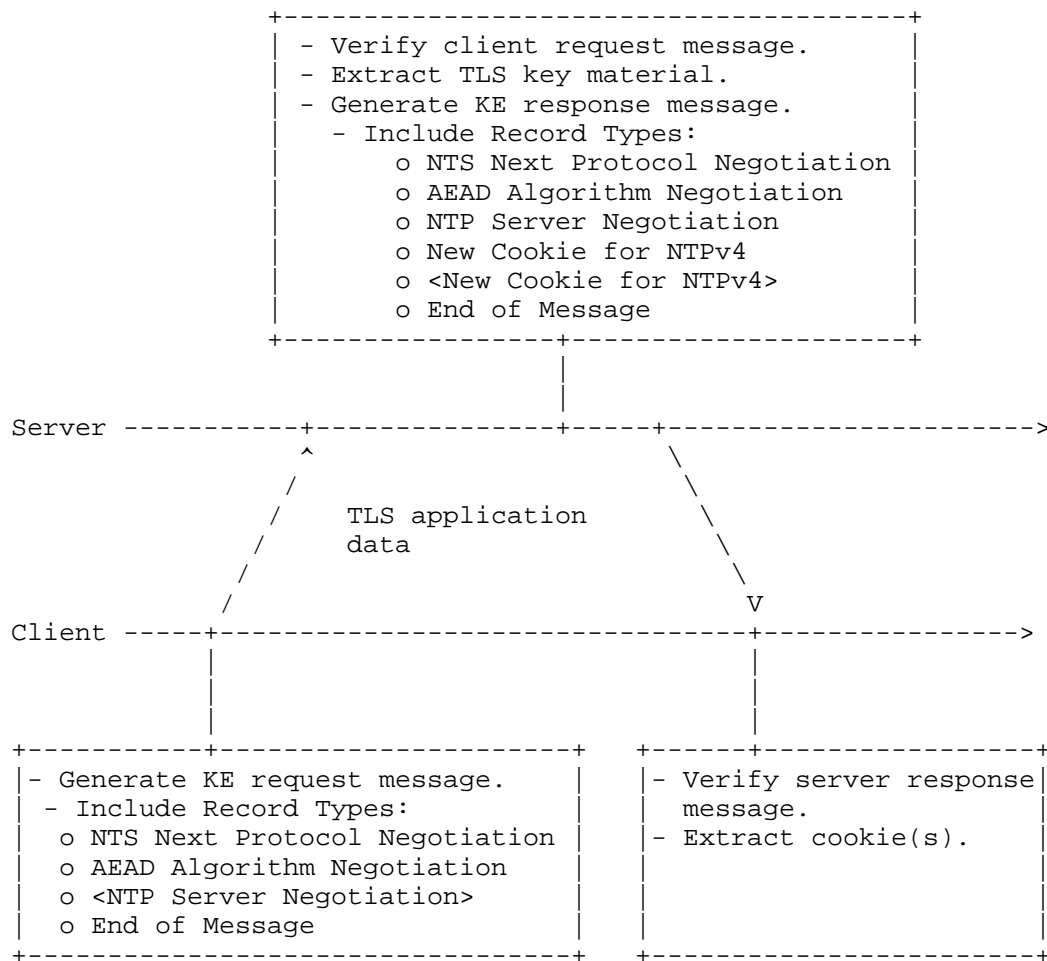


Figure 3: NTS Key Exchange Messages

4.1. NTS-KE Record Types

The following NTS-KE Record Types are defined:

4.1.1. End of Message

The End of Message record has a Record Type number of 0 and a zero-length body. It MUST occur exactly once as the final record of every NTS-KE request and response. The Critical Bit MUST be set.

4.1.2. NTS Next Protocol Negotiation

The NTS Next Protocol Negotiation record has a Record Type number of 1. It MUST occur exactly once in every NTS-KE request and response. Its body consists of a sequence of 16-bit unsigned integers in network byte order. Each integer represents a Protocol ID from the IANA Network Time Security Next Protocols registry. The Critical Bit MUST be set.

The Protocol IDs listed in the client's NTS Next Protocol Negotiation record denote those protocols which the client wishes to speak using the key material established through this NTS-KE session. The Protocol IDs listed in the server's response MUST comprise a subset of those listed in the request and denote those protocols which the server is willing and able to speak using the key material established through this NTS-KE 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.

4.1.3. Error

The Error record has a Record Type number of 2. Its body is exactly two octets long, consisting of an unsigned 16-bit integer in network byte order, denoting an error code. The Critical Bit MUST be set.

Clients MUST NOT include Error records in their request. If clients receive a server response which includes an Error record, they MUST discard any negotiated key material and MUST NOT proceed to the Next Protocol.

The following error codes are defined:

Error code 0 means "Unrecognized Critical Record". The server MUST respond with this error code if the request included a record which the server did not understand and which had its Critical Bit set. The client SHOULD NOT retry its request without modification.

Error code 1 means "Bad Request". The server MUST respond with this error if, upon the expiration of an implementation-defined timeout, it has not yet received a complete and syntactically well-formed request from the client.

4.1.4. Warning

The Warning record has a Record Type number of 3. Its body is exactly two octets long, consisting of an unsigned 16-bit integer in

network byte order, denoting a warning code. The Critical Bit MUST be set.

Clients MUST NOT include Warning records in their request. If clients receive a server response which includes a Warning record, they MAY discard any negotiated key material and abort without proceeding to the Next Protocol. Unrecognized warning codes MUST be treated as errors.

This memo defines no warning codes.

4.1.5. AEAD Algorithm Negotiation

The AEAD Algorithm Negotiation record has a Record Type number of 4. Its body consists of a sequence of unsigned 16-bit integers in network byte order, denoting Numeric Identifiers from the IANA AEAD registry [RFC5116]. The Critical Bit MAY be set.

If the NTS Next Protocol Negotiation record offers Protocol ID 0 (for NTPv4), then this record MUST be included exactly once. Other protocols MAY require it as well.

When included in a request, this record denotes which AEAD algorithms the client is willing to use to secure the Next Protocol, in decreasing preference order. When included in a response, this record denotes which algorithm the server chooses to use. It is empty if the server supports none of the algorithms offered. In requests, the list MUST include at least one algorithm. In responses, it MUST include at most one. Honoring the client's preference order is OPTIONAL: servers may select among any of the client's offered choices, even if they are able to support some other algorithm which the client prefers more.

Server implementations of NTS extension fields for NTPv4 (Section 5) MUST support AEAD_AES_SIV_CMACE_256 [RFC5297] (Numeric Identifier 15). That is, if the client includes AEAD_AES_SIV_CMACE_256 in its AEAD Algorithm Negotiation record and the server accepts Protocol ID 0 (NTPv4) in its NTS Next Protocol Negotiation record, then the server's AEAD Algorithm Negotiation record MUST NOT be empty.

4.1.6. New Cookie for NTPv4

The New Cookie for NTPv4 record has a Record Type number of 5. The contents of its body SHALL be implementation-defined and clients MUST NOT attempt to interpret them. See Section 7 for a suggested construction.

Clients MUST NOT send records of this type. Servers MUST send at least one record of this type, and SHOULD send eight of them, if the Next Protocol Negotiation response record contains Protocol ID 0 (NTPv4) and the AEAD Algorithm Negotiation response record is not empty. The Critical Bit SHOULD NOT be set.

4.1.7. NTP Server Negotiation

The NTP Server Negotiation record has a Record Type number of 6. The record MAY be sent by a client in a request and SHOULD be sent by a server as part of a reply. Its body consists of a sequence of IPv4 and/or IPv6 addresses. Both address types are represented by 16 octets in network byte order. To achieve this, IPv4 addresses are represented as "IPv4-mapped IPv6 addresses" in the format specified in RFC 4291, Section 2.5.5.2 [RFC4291]. For example: The IPv4 address 192.0.2.1 would be mapped to the IPv6 address space as ::ffff:192.0.2.1. The Critical Bit SHOULD be set.

When used in a request, the NTP Server Negotiation record is the client's way of indicating to the KE server which NTP servers it wishes to receive cookies for. Honoring the client's NTP server preferences is OPTIONAL. When used in a response, this record informs the client about which NTP servers the received cookies can be used with in the next phase of the protocol. The client SHOULD NOT attempt to use the received cookies with any other NTP servers than those indicated by the KE server.

If a response does not include this record, the client SHOULD assume that the received cookies are valid for use with an NTP server at the same network address as the key exchange server.

4.2. Key Extraction (generally)

Following a successful run of the NTS-KE protocol, key material SHALL be extracted according to RFC 5705 [RFC5705]. Inputs to the exporter function are to be constructed in a manner specific to the negotiated Next Protocol. However, all protocols which utilize NTS-KE MUST conform to the following two rules:

The disambiguating label string MUST be "EXPORTER-network-time-security/1".

The per-association context value MUST be provided and MUST begin with the two-octet Protocol ID which was negotiated as a Next Protocol.

4.3. Key Extraction (for NTPv4)

Following a successful run of the NTS-KE protocol wherein Protocol ID 0 (NTPv4) is selected as a Next Protocol, two AEAD keys SHALL be extracted: a client-to-server (C2S) key and a server-to-client (S2C) key. These keys SHALL be computed according to RFC 5705 [RFC5705], using the following inputs.

The disambiguating label string SHALL be "EXPORTER-network-time-security/1".

The per-association context value SHALL consist of the following five octets:

The first two octets SHALL be zero (the Protocol ID for NTPv4).

The next two octets SHALL be the Numeric Identifier of the negotiated AEAD Algorithm in network byte order.

The final octet SHALL be 0x00 for the C2S key and 0x01 for the S2C key.

Implementations wishing to derive additional keys for private or experimental use MUST NOT do so by extending the above-specified syntax for per-association context values. Instead, they SHOULD use their own disambiguating label string. Note that RFC 5705 [RFC5705] provides that disambiguating label strings beginning with "EXPERIMENTAL" MAY be used without IANA registration.

5. NTS Extension Fields for NTPv4

5.1. Packet Structure Overview

In general, an NTS-protected NTPv4 packet consists of:

The usual 48-octet NTP header which is authenticated but not encrypted.

Some extension fields which are authenticated but not encrypted.

An extension field which contains AEAD output (i.e., an authentication tag and possible ciphertext). The corresponding plaintext, if non-empty, consists of some extension fields which benefit from both encryption and authentication.

Possibly, some additional extension fields which are neither encrypted nor authenticated. These are discarded by the receiver.

Always included among the authenticated or authenticated-and-encrypted extension fields are a cookie extension field and a unique identifier extension field. The purpose of the cookie extension field is to enable the server to offload storage of session state onto the client. The purpose of the unique identifier extension field is to protect the client from replay attacks.

5.2. The Unique Identifier Extension Field

The Unique Identifier extension field provides the client with a cryptographically strong means of detecting replayed packets. It has a Field Type of [[TBD2]]. When the extension field is included in a client packet (mode 3), its body SHALL consist of a string of octets generated uniformly at random. The string MUST be at least 32 octets long. When the extension field is included in a server packet (mode 4), its body SHALL contain the same octet string as was provided in the client packet to which the server is responding. All server packets generated by NTS-implementing servers in response to client packets containing this extension field MUST also contain this field with the same content as in the client's request. The field's use in modes other than client-server is not defined.

This extension field MAY also be used standalone, without NTS, in which case it provides the client with a means of detecting spoofed packets from off-path attackers. Historically, NTP's origin timestamp field has played both these roles, but for cryptographic purposes this is suboptimal because it is only 64 bits long and, depending on implementation details, most of those bits may be predictable. In contrast, the Unique Identifier extension field enables a degree of unpredictability and collision resistance more consistent with cryptographic best practice.

5.3. The NTS Cookie Extension Field

The NTS Cookie extension field has a Field Type of [[TBD3]]. Its purpose is to carry information which enables the server to recompute keys and other session state without having to store any per-client state. The contents of its body SHALL be implementation-defined and clients MUST NOT attempt to interpret them. See Section 7 for a suggested construction. The NTS Cookie extension field MUST NOT be included in NTP packets whose mode is other than 3 (client) or 4 (server).

5.4. The NTS Cookie Placeholder Extension Field

The NTS Cookie Placeholder extension field has a Field Type of [[TBD4]]. When this extension field is included in a client packet (mode 3), it communicates to the server that the client wishes it to

send additional cookies in its response. This extension field MUST NOT be included in NTP packets whose mode is other than 3.

Whenever an NTS Cookie Placeholder extension field is present, it MUST be accompanied by an NTS Cookie extension field. The body length of the NTS Cookie Placeholder extension field MUST be the same as the body length of the NTS Cookie extension field. This length requirement serves to ensure that the response will not be larger than the request, in order to improve timekeeping precision and prevent DDoS amplification. The contents of the NTS Cookie Placeholder extension field's body are undefined and, aside from checking its length, MUST be ignored by the server.

5.5. The NTS Authenticator and Encrypted Extension Fields Extension Field

The NTS Authenticator and Encrypted Extension Fields extension field is the central cryptographic element of an NTS-protected NTP packet. Its Field Type is [[TBD5]]. It SHALL be formatted according to Figure 4 and include the following fields:

Nonce length: Two octets in network byte order, giving the length of the Nonce field, excluding any padding, interpreted as an unsigned integer.

Ciphertext Length: Two octets in network byte order, giving the length of the Ciphertext field, excluding any padding, interpreted as an unsigned integer.

Nonce: A nonce as required by the negotiated AEAD Algorithm. The field is zero-padded to a word (four octets) boundary.

Ciphertext: The output of the negotiated AEAD Algorithm. The structure of this field is determined by the negotiated algorithm, but it typically contains an authentication tag in addition to the actual ciphertext. The field is zero-padded to a word (four octets) boundary.

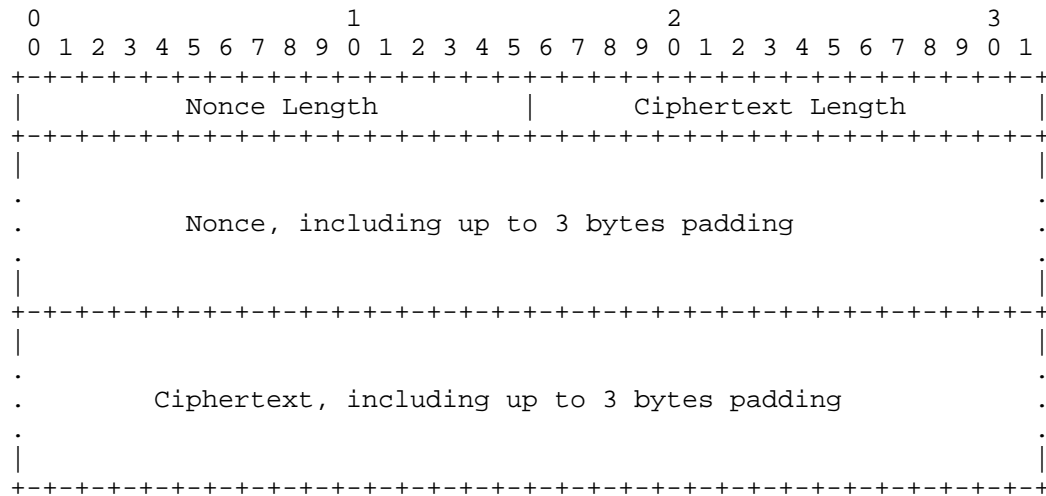


Figure 4: NTS Authenticator and Encrypted Extension Fields Extension Field Format

The Ciphertext field SHALL be formed by providing the following inputs to the negotiated AEAD Algorithm:

K: For packets sent from the client to the server, the C2S key SHALL be used. For packets sent from the server to the client, the S2C key SHALL be used.

A: The associated data SHALL consist of the portion of the NTP packet beginning from the start of the NTP header and ending at the end of the last extension field which precedes the NTS Authenticator and Encrypted Extension Fields extension field.

P: The plaintext SHALL consist of all (if any) NTP extension fields to be encrypted. The format of any such fields SHALL be in accordance with RFC 7822 [RFC7822]. If multiple extension fields are present they SHALL be joined by concatenation.

N: The nonce SHALL be formed however required by the negotiated AEAD Algorithm.

The NTS Authenticator and Encrypted Extension Fields extension field MUST NOT be included in NTP packets whose mode is other than 3 (client) or 4 (server).

6. Protocol Details

A client sending an NTS-protected request SHALL include the following extension fields as displayed in Figure 5:

Exactly one Unique Identifier extension field which MUST be authenticated, MUST NOT be encrypted, and whose contents MUST NOT duplicate those of any previous request.

Exactly one NTS Cookie extension field which MUST be authenticated and MUST NOT be encrypted. The cookie MUST be one which has been previously provided to the client; either from the key exchange server during the NTS-KE handshake or from the NTP server in response to a previous NTS-protected NTP request. To protect the client's privacy, the same cookie SHOULD NOT be included in multiple requests. If the client does not have any cookies that it has not already sent, it SHOULD initiate a re-run the NTS-KE protocol.

Exactly one NTS Authenticator and Encrypted Extension Fields extension field, generated using an AEAD Algorithm and C2S key established through NTS-KE.

The client MAY include one or more NTS Cookie Placeholder extension fields which MUST be authenticated and MAY be encrypted. The number of NTS Cookie Placeholder extension fields that the client includes SHOULD be such that if the client includes N placeholders and the server sends back N+1 cookies, the number of unused cookies stored by the client will come to eight. The client SHOULD NOT include more than seven NTS Cookie Placeholder extension fields in a request. When both the client and server adhere to all cookie-management guidance provided in this memo, the number of placeholder extension fields will equal the number of dropped packets since the last successful volley.

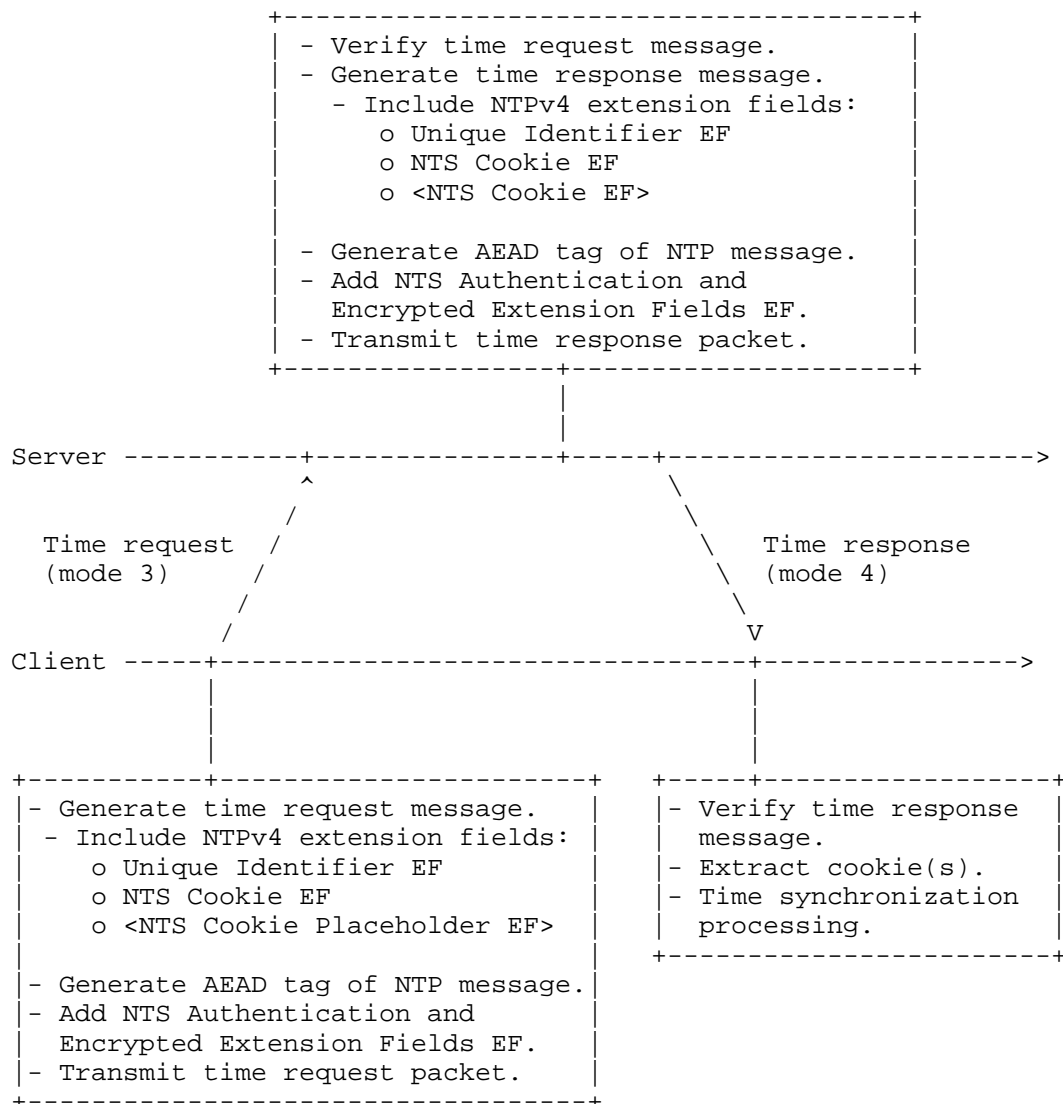


Figure 5: NTS Time Synchronization Messages

The client MAY include additional (non-NTS-related) extension fields which MAY appear prior to the NTS Authenticator and Encrypted Extension Fields extension fields (therefore authenticated but not encrypted), within it (therefore encrypted and authenticated), or after it (therefore neither encrypted nor authenticated). In general, however, the server MUST discard any unauthenticated extension fields and process the packet as though they were not

present. Servers MAY implement exceptions to this requirement for particular extension fields if their specification explicitly provides for such.

Upon receiving an NTS-protected request, the server SHALL (through some implementation-defined mechanism) use the cookie to recover the AEAD Algorithm, C2S key, and S2C key associated with the request, and then use the C2S key to authenticate the packet and decrypt the ciphertext. If the cookie is valid and authentication and decryption succeed, the server SHALL include the following extension fields in its response:

Exactly one Unique Identifier extension field which MUST be authenticated, MUST NOT be encrypted, and whose contents SHALL echo those provided by the client.

Exactly one NTS Authenticator and Encrypted Extension Fields extension field, generated using the AEAD algorithm and S2C key recovered from the cookie provided by the client.

One or more NTS Cookie extension fields which MUST be authenticated and encrypted. The number of NTS Cookie extension fields included SHOULD be equal to, and MUST NOT exceed, one plus the number of valid NTS Cookie Placeholder extension fields included in the request. The cookies returned in those fields MUST be valid for use with the NTP server that sent them. They MAY be valid for other NTP servers as well, but there is no way for the server to indicate this.

We emphasize the contrast that NTS Cookie extension fields MUST NOT be encrypted when sent from client to server, but MUST be encrypted from sent from server to client. The former is necessary in order for the server to be able to recover the C2S and S2C keys, while the latter is necessary to satisfy the unlinkability goals discussed in Section 11.1. We emphasize also that "encrypted" means encapsulated within the the NTS Authenticator and Encrypted Extensions extension field. While the body of an NTS Cookie extension field will generally consist of some sort of AEAD output (regardless of whether the recommendations of Section 7 are precisely followed), this is not sufficient to make the extension field "encrypted".

The server MAY include additional (non-NTS-related) extension fields which MAY appear prior to the NTS Authenticator and Encrypted Extension Fields extension field (therefore authenticated but not encrypted), within it (therefore encrypted and authenticated), or after it (therefore neither encrypted nor authenticated). In general, however, the client MUST discard any unauthenticated extension fields and process the packet as though they were not

present. Clients MAY implement exceptions to this requirement for particular extension fields if their specification explicitly provides for such.

Upon receiving an NTS-protected response, the client MUST verify that the Unique Identifier matches that of an outstanding request, and that the packet is authentic under the S2C key associated with that request. If either of these checks fails, the packet MUST be discarded without further processing.

If the server is unable to validate the cookie or authenticate the request, it SHOULD respond with a Kiss-o'-Death (KoD) packet (see RFC 5905, Section 7.4 [RFC5905]) with kiss code "NTSN", meaning "NTS negative-acknowledgment (NAK)". It MUST NOT include any NTS Cookie or NTS Authenticator and Encrypted Extension Fields extension fields.

If the NTP server has previously responded with authentic NTS-protected NTP packets (i.e., packets containing the NTS Authenticator and Encrypted Extension Fields extension field), the client MUST verify that any KoD packets received from the server contain the Unique Identifier extension field and that the Unique Identifier matches that of an outstanding request. If this check fails, the packet MUST be discarded without further processing. If this check passes, the client MUST comply with RFC 5905, Section 7.4 [RFC5905] where required. A client MAY automatically re-run the NTS-KE protocol upon forced disassociation from an NTP server. In that case, it MUST be able to detect and stop looping between the NTS-KE and NTP servers.

Upon reception of the NTS NAK kiss code, the client SHOULD wait until the next poll for a valid NTS-protected response and if none is received, initiate a fresh NTS-KE handshake to try to renegotiate new cookies, AEAD keys, and parameters. If the NTS-KE handshake succeeds, the client MUST discard all old cookies and parameters and use the new ones instead. As long as the NTS-KE handshake has not succeeded, the client SHOULD continue polling the NTP server using the cookies and parameters it has.

The client MAY reuse cookies in order to prioritize resilience over unlinkability. Which of the two that should be prioritized in any particular case is dependent on the application and the user's preference. Section 11.1 describes the privacy considerations of this in further detail.

To allow for NTP session restart when the NTS-KE server is unavailable and to reduce NTS-KE server load, the client SHOULD keep at least one unused but recent cookie, AEAD keys, negotiated AEAD algorithm, and other necessary parameters on persistent storage.

This way, the client is able to resume the NTP session without performing renewed NTS-KE negotiation.

7. Suggested Format for NTS Cookies

This section is non-normative. It gives a suggested way for servers to construct NTS cookies. All normative requirements are stated in Section 4.1.6 and Section 5.3.

The role of cookies in NTS is closely analogous to that of session cookies in TLS. Accordingly, the thematic resemblance of this section to RFC 5077 [RFC5077] is deliberate and the reader should likewise take heed of its security considerations.

Servers should select an AEAD algorithm which they will use to encrypt and authenticate cookies. The chosen algorithm should be one such as AEAD_AES_SIV_CMAC_256 [RFC5297] which resists accidental nonce reuse. It need not be the same as the one that was negotiated with the client. Servers should randomly generate and store a master AEAD key 'K'. Servers should additionally choose a non-secret, unique value 'I' as key-identifier for 'K'.

Servers should periodically (e.g., once daily) generate a new pair (I,K) and immediately switch to using these values for all newly-generated cookies. Immediately following each such key rotation, servers should securely erase any keys generated two or more rotation periods prior. Servers should continue to accept any cookie generated using keys that they have not yet erased, even if those keys are no longer current. Erasing old keys provides for forward secrecy, limiting the scope of what old information can be stolen if a master key is somehow compromised. Holding on to a limited number of old keys allows clients to seamlessly transition from one generation to the next without having to perform a new NTS-KE handshake.

The need to keep keys synchronized between NTS-KE and NTP servers as well as across load-balanced clusters can make automatic key rotation challenging. However, the task can be accomplished without the need for central key-management infrastructure by using a ratchet, i.e., making each new key a deterministic, cryptographically pseudo-random function of its predecessor. A recommended concrete implementation of this approach is to use HKDF [RFC5869] to derive new keys, using the key's predecessor as Input Keying Material and its key identifier as a salt.

To form a cookie, servers should first form a plaintext 'P' consisting of the following fields:

The AEAD algorithm negotiated during NTS-KE.

The S2C key.

The C2S key.

Servers should then generate a nonce 'N' uniformly at random, and form AEAD output 'C' by encrypting 'P' under key 'K' with nonce 'N' and no associated data.

The cookie should consist of the tuple '(I,N,C)'.

To verify and decrypt a cookie provided by the client, first parse it into its components 'I', 'N', and 'C'. Use 'I' to look up its decryption key 'K'. If the key whose identifier is 'I' has been erased or never existed, decryption fails; reply with an NTS NAK. Otherwise, attempt to decrypt and verify ciphertext 'C' using key 'K' and nonce 'N' with no associated data. If decryption or verification fails, reply with an NTS NAK. Otherwise, parse out the contents of the resulting plaintext 'P' to obtain the negotiated AEAD algorithm, S2C key, and C2S key.

8. Usage of NTP pools

Many NTP server pools exist. Some of them have thousands of individual servers spread out over several continents. Due to their size and prevalence, it can be expected that a significant portion of NTP users are users of NTP pools.

The separation of the initial NTS key exchange from the authenticated NTP protocol simplifies the implementation of NTS on pool infrastructures. Since NTS key exchange over TLS is expected to be a rare occurrence in comparison with the normal authenticated NTP request and response traffic, even large pools should require a relatively small number of NTS-KE servers. This eliminates the need for complex certificate infrastructures. The lower timing and hardware requirements on NTS-KE servers also provide for load-balancing solutions that aren't suitable for NTP servers, such as virtual machine implementations that are started and stopped as needed.

The ability for NTS-KE servers to freely choose what NTP servers they will issue cookies for means that each pool can implement whatever secret-sharing system between NTS-KE and NTP servers it deems suitable. For example, in a large pool where the trust in the individual NTP server administrators is relatively low, it may be necessary to have separate shared secrets for each possible pair of NTS-KE and NTP servers. It should also be noted that not all NTS-KE

servers in a pool must have the ability to issue cookies for all NTP servers in that pool.

Due to their freedom to choose what servers to issue cookies for, NTS-KE servers can perform a number of functions in addition to authenticating themselves to clients and issuing cookies. This includes load-balancing and geographic assignment of clients to NTP servers.

9. IANA Considerations

9.1. Service Name and Transport Protocol Port Number Registry

IANA is requested to allocate two entries, identical except for the Transport Protocol, in the Service Name and Transport Protocol Port Number Registry [RFC6335] as follows:

Service Name: nts

Transport Protocol: tcp, udp

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Network Time Security

Reference: [[this memo]]

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

9.2. TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs Registry

IANA is requested to allocate the following entry in the TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs registry [RFC7301]:

Protocol: Network Time Security Key Establishment, version 1

Identification Sequence:

0x6E 0x74 0x73 0x6B 0x65 0x2F 0x31 ("ntske/1")

Reference: [[this memo]], Section 4

9.3. TLS Exporter Labels Registry

IANA is requested to allocate the following entry in the TLS Exporter Labels Registry [RFC5705]:

Value	DTLS-OK	Recommended	Reference	Note
EXPORTER-network-time-security/1	Y	Y	[[this memo]], Section 4.2	

9.4. NTP Kiss-o'-Death Codes Registry

IANA is requested to allocate the following entry in the registry of NTP Kiss-o'-Death Codes [RFC5905]:

Code	Meaning	Reference
NTSN	Network Time Security (NTS) negative-acknowledgment (NAK)	[[this memo]], Section 6

9.5. NTP Extension Field Types Registry

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

Field Type	Meaning	Reference
[[TBD2]]	Unique Identifier	[[this memo]], Section 5.2
[[TBD3]]	NTS Cookie	[[this memo]], Section 5.3
[[TBD4]]	NTS Cookie Placeholder	[[this memo]], Section 5.4
[[TBD5]]	NTS Authenticator and Encrypted Extension Fields	[[this memo]], Section 5.5

9.6. Network Time Security Key Establishment Record Types Registry

IANA is requested to create a new registry entitled "Network Time Security Key Establishment Record Types". Entries SHALL have the following fields:

Record Type Number (REQUIRED): An integer in the range 0-32767 inclusive.

Description (REQUIRED): A short text description of the purpose of the field.

Set Critical Bit (REQUIRED): One of "MUST", "SHOULD", "MAY", "SHOULD NOT", or "MUST NOT".

Reference (REQUIRED): A reference to a document specifying the semantics of the record.

The policy for allocation of new entries in this registry SHALL vary by the Record Type Number, as follows:

0-1023: IETF Review.

1024-16383: Specification Required.

16384-32767: Private and Experimental Use.

Applications for new entries SHALL specify the contents of the Description, Set Critical Bit, and Reference fields as well as which of the above ranges the Record Type Number should be allocated from. Applicants MAY request a specific Record Type Number and such requests MAY be granted at the registrar's discretion.

The initial contents of this registry SHALL be as follows:

Record Type Number	Description	Set Critical Bit	Reference
0	End of Message	MUST	[[this memo]], Section 4.1.1
1	NTS Next Protocol Negotiation	MUST	[[this memo]], Section 4.1.2
2	Error	MUST	[[this memo]], Section 4.1.3
3	Warning	MUST	[[this memo]], Section 4.1.4
4	AEAD Algorithm Negotiation	MAY	[[this memo]], Section 4.1.5
5	New Cookie for NTPv4	SHOULD NOT	[[this memo]], Section 4.1.6
6	NTP Server Negotiation	SHOULD	[[this memo]], Section 4.1.7
16384-32767	Reserved for Private & Experimental Use	MAY	[[this memo]]

9.7. Network Time Security Next Protocols Registry

IANA is requested to create a new registry entitled "Network Time Security Next Protocols". Entries SHALL have the following fields:

Protocol ID (REQUIRED): An integer in the range 0-65535 inclusive, functioning as an identifier.

Protocol Name (REQUIRED): A short text string naming the protocol being identified.

Reference (RECOMMENDED): A reference to a relevant specification document. If no relevant document exists, a point-of-contact for questions regarding the entry SHOULD be listed here in lieu.

Applications for new entries in this registry SHALL specify all desired fields and SHALL be granted upon approval by a Designated Expert. Protocol IDs 32768-65535 SHALL be reserved for Private or Experimental Use and SHALL NOT be registered.

The initial contents of this registry SHALL be as follows:

Protocol ID	Protocol Name	Reference
0	Network Time Protocol version 4 (NTPv4)	[[this memo]]
32768-65535	Reserved for Private or Experimental Use	Reserved by [[this memo]]

9.8. Network Time Security Error and Warning Codes Registries

IANA is requested to create two new registries entitled "Network Time Security Error Codes" and "Network Time Security Warning Codes". Entries in each SHALL have the following fields:

Number (REQUIRED): An integer in the range 0-65535 inclusive

Description (REQUIRED): A short text description of the condition.

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

The policy for allocation of new entries in these registries SHALL vary by their Number, as follows:

0-1023: IETF Review.

1024-32767: Specification Required.

32768-65535: Private and Experimental Use.

The initial contents of the Network Time Security Error Codes Registry SHALL be as follows:

Number	Description	Reference
0	Unrecognized Critical Extension	[[this memo]], Section 4.1.3
1	Bad Request	[[this memo]], Section 4.1.3

The Network Time Security Warning Codes Registry SHALL initially be empty.

10. Security Considerations

10.1. Sensitivity to DDoS attacks

The introduction of NTS brings with it the introduction of asymmetric cryptography to NTP. Asymmetric cryptography is necessary for initial server authentication and AEAD key extraction. Asymmetric cryptosystems are generally orders of magnitude slower than their symmetric counterparts. This makes it much harder to build systems that can serve requests at a rate corresponding to the full line speed of the network connection. This, in turn, opens up a new possibility for DDoS attacks on NTP services.

The main protection against these attacks in NTS lies in that the use of asymmetric cryptosystems is only necessary in the initial NTS-KE phase of the protocol. Since the protocol design enables separation of the NTS-KE and NTP servers, a successful DDoS attack on an NTS-KE server separated from the NTP service it supports will not affect NTP users that have already performed initial authentication, AEAD key extraction, and cookie exchange. Furthermore, as noted in Section 8, NTP-KE capacity is easier to scale up and down than NTP server capacity.

NTS users should also consider that they are not fully protected against DDoS attacks by on-path adversaries. In addition to dropping packets and attacks such as those described in Section 10.4, an on-path attacker can send spoofed kiss-o'-death replies, which are not authenticated, in response to NTP requests. This could result in significantly increased load on the NTS-KE server. Implementers have to weigh the user's need for unlinkability against the added resilience that comes with cookie reuse in cases of NTS-KE server unavailability.

10.2. Avoiding DDoS Amplification

Certain non-standard and/or deprecated features of the Network Time Protocol enable clients to send a request to a server which causes the server to send a response much larger than the request. Servers which enable these features can be abused in order to amplify traffic volume in DDoS attacks by sending them a request with a spoofed source IP. In recent years, attacks of this nature have become an endemic nuisance.

NTS is designed to avoid contributing any further to this problem by ensuring that NTS-related extension fields included in server responses will be the same size as the NTS-related extension fields sent by the client. In particular, this is why the client is required to send a separate and appropriately padded-out NTS Cookie

Placeholder extension field for every cookie it wants to get back, rather than being permitted simply to specify a desired quantity.

Due to the RFC 7822 [RFC7822] requirement that extensions be padded and aligned to four-octet boundaries, response size may still in some cases exceed request size by up to three octets. This is sufficiently inconsequential that we have declined to address it.

10.3. Initial Verification of Server Certificates

NTS's security goals are undermined if the client fails to verify that the X.509 certificate chain presented by the NTS-KE server is valid and rooted in a trusted certificate authority. RFC 5280 [RFC5280] and RFC 6125 [RFC6125] specify how such verification is to be performed in general. However, the expectation that the client does not yet have a correctly-set system clock at the time of certificate verification presents difficulties with verifying that the certificate is within its validity period, i.e., that the current time lies between the times specified in the certificate's notBefore and notAfter fields. It may be operationally necessary in some cases for a client to accept a certificate which appears to be expired or not yet valid. While there is no perfect solution to this problem, there are several mitigations the client can implement to make it more difficult for an adversary to successfully present an expired certificate:

Check whether the system time is in fact unreliable. If the system clock has previously been synchronized since last boot, then on operating systems which implement a kernel-based phase-locked-loop API, a call to `ntp_gettime()` should show a maximum error less than `NTP_PHASE_MAX`. In this case, the clock SHOULD be considered reliable and certificates can be strictly validated.

Allow the system administrator to specify that certificates should **always** be strictly validated. Such a configuration is appropriate on systems which have a battery-backed clock and which can reasonably prompt the user to manually set an approximately-correct time if it appears to be needed.

Once the clock has been synchronized, periodically write the current system time to persistent storage. Do not accept any certificate whose notAfter field is earlier than the last recorded time.

Do not process time packets from servers if the time computed from them falls outside the validity period of the server's certificate.

Use multiple time sources. The ability to pass off an expired certificate is only useful to an adversary who has compromised the corresponding private key. If the adversary has compromised only a minority of servers, NTP's selection algorithm (RFC 5905 section 11.2.1 [RFC5905]) will protect the client from accepting bad time from the adversary-controlled servers.

10.4. Delay Attacks

In a packet delay attack, an adversary with the ability to act as a man-in-the-middle delays time synchronization packets between client and server asymmetrically [RFC7384]. Since NTP's formula for computing time offset relies on the assumption that network latency is roughly symmetrical, this leads to the client to compute an inaccurate value [Mizrahi]. The delay attack does not reorder or modify the content of the exchanged synchronization packets. Therefore, cryptographic means do not provide a feasible way to mitigate this attack. However, the maximum error that an adversary can introduce is bounded by half of the round trip delay.

RFC 5905 [RFC5905] specifies a parameter called MAXDIST which denotes the maximum round-trip latency (including not only the immediate round trip between client and server, but the whole distance back to the reference clock as reported in the Root Delay field) that a client will tolerate before concluding that the server is unsuitable for synchronization. The standard value for MAXDIST is one second, although some implementations use larger values. Whatever value a client chooses, the maximum error which can be introduced by a delay attack is MAXDIST/2.

Usage of multiple time sources, or multiple network paths to a given time source [Shpiner], may also serve to mitigate delay attacks if the adversary is in control of only some of the paths.

10.5. Random Number Generation

At various points in NTS, the generation of cryptographically secure random numbers is required. Whenever this draft specifies the use of random numbers, cryptographically secure random number generation MUST be used. RFC 4086 [RFC4086] contains guidelines concerning this topic.

11. Privacy Considerations

11.1. Unlinkability

Unlinkability prevents a device from being tracked when it changes network addresses (e.g., because said device moved between different networks). In other words, unlinkability thwarts an attacker that seeks to link a new network address used by a device with a network address that it was formerly using through recognizable data that the device persistently sends as part of an NTS-secured NTP association. This is the justification for continually supplying the client with fresh cookies, so that a cookie never represents recognizable data in the sense outlined above.

NTS's unlinkability objective is merely to not leak any additional data that could be used to link a device's network address. NTS does not rectify legacy linkability issues that are already present in NTP. Thus, a client that requires unlinkability must also minimize information transmitted in a client query (mode 3) packet as described in the NTP Client Data Minimization Internet-Draft [I-D.ietf-ntp-data-minimization].

The unlinkability objective only holds for time synchronization traffic, as opposed to key exchange traffic. This implies that it cannot be guaranteed for devices that function not only as time clients, but also as time servers (because the latter can be externally triggered to send authentication data).

It should also be noted that it could be possible to link devices that operate as time servers from their time synchronization traffic, using information exposed in (mode 4) server response packets (e.g., reference ID, reference time, stratum, poll). Also, devices that respond to NTP control queries could be linked using the information revealed by control queries.

11.2. Confidentiality

NTS does not protect the confidentiality of information in NTP's header fields. When clients implement NTP Client Data Minimization [I-D.ietf-ntp-data-minimization], client packet headers do not contain any information which the client could conceivably wish to keep secret: one field is random and all others are fixed. Information in server packet headers is likewise public: the origin timestamp is copied from the client's (random) transmit timestamp and all other fields are set the same regardless of the identity of the client making the request.

Future extension fields could hypothetically contain sensitive information, in which case NTS provides a mechanism for encrypting them.

12. Acknowledgements

The authors would like to thank Richard Barnes, Steven Bellovin, Scott Fluhrer, Sharon Goldberg, Russ Housley, Martin Langer, Miroslav Lichvar, Aanchal Malhotra, Dave Mills, Danny Mayer, Karen O'Donoghue, Eric K. Rescorla, Stephen Roettger, Kurt Roeckx, Kyle Rose, Rich Salz, Brian Sniffen, Susan Sons, Douglas Stebila, Harlan Stenn, Martin Thomson, and Richard Welty for contributions to this document and comments on the design of NTS.

The idea of separation of the NTS-KE server from the NTP server was added by Marcus Dansarie and Ragnar Sundblad. Thanks for this work goes to Patrik Faltstroem (Faltstrom) and Joachim Stroembergsson (Strombergsson) for review and ideas.

13. References

13.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [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>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [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>>.
- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <<https://www.rfc-editor.org/info/rfc5705>>.
- [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>>.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [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>>.
- [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>>.
- [RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", RFC 7507, DOI 10.17487/RFC7507, April 2015, <<https://www.rfc-editor.org/info/rfc7507>>.
- [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>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, July 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

13.2. Informative References

- [I-D.ietf-ntp-data-minimization] Franke, D. and A. Malhotra, "NTP Client Data Minimization", draft-ietf-ntp-data-minimization-02 (work in progress), July 2018.
- [Mizrahi] Mizrahi, T., "A game theoretic analysis of delay attacks against time synchronization protocols", in Proceedings of Precision Clock Synchronization for Measurement Control and Communication, ISPCS 2012, pp. 1-6, September 2012.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.

- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <<https://www.rfc-editor.org/info/rfc5077>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [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>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [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>>.
- [Shpiner] "Multi-path Time Protocols", in Proceedings of IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS), September 2013.

Appendix A. Terms and Abbreviations

AEAD	Authenticated Encryption with Associated Data [RFC5116]
ALPN	Application-Layer Protocol Negotiation [RFC7301]
C2S	Client-to-server
DDoS	Distributed Denial-of-Service
EF	Extension Field [RFC5905]
HKDF	Hashed Message Authentication Code-based Key Derivation Function [RFC5869]
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol

KoD Kiss-o'-Death [RFC5905]
NTP Network Time Protocol [RFC5905]
NTS Network Time Security
NTS-KE Network Time Security Key Exchange
S2C Server-to-client
SCSV Signaling Cipher Suite Value [RFC7507]
TCP Transmission Control Protocol [RFC0793]
TLS Transport Layer Security [RFC8446]
UDP User Datagram Protocol [RFC0768]

Authors' Addresses

Daniel Fox Franke

Email: dfoxfranke@gmail.com

URI: <https://www.dfranke.us>

Dieter Sibold
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8420
Fax: +49-531-592-698420
Email: dieter.sibold@ptb.de

Kristof Teichel
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-4471
Email: kristof.teichel@ptb.de

Marcus Dansarie

Email: marcus@dansarie.se

Ragnar Sundblad
Netnod

Email: ragge@netnod.se

Internet Engineering Task Force
Internet-Draft
Intended status: Best Current Practice
Expires: September 27, 2019

D. Reilly, Ed.
Orolia USA
H. Stenn
Network Time Foundation
D. Sibold
PTB
March 26, 2019

Network Time Protocol Best Current Practices
draft-ietf-ntp-bcp-13

Abstract

The Network Time Protocol (NTP) is one of the oldest protocols on the Internet and has been widely used since its initial publication. This document is a collection of Best Practices for general operation of NTP servers and clients on the Internet. It includes recommendations for stable, accurate and secure operation of NTP infrastructure. This document is targeted at NTP version 4 as described in RFC 5905.

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 27, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. General Network Security Best Practices	3
2.1. BCP 38	3
3. NTP Configuration Best Practices	4
3.1. Keeping NTP up to date	4
3.2. Use enough time sources	4
3.3. Use a diversity of Reference Clocks	5
3.4. Control Messages	6
3.5. Monitoring	7
3.6. Using Pool Servers	7
3.7. Leap Second Handling	8
3.7.1. Leap Smearing	9
4. NTP Security Mechanisms	10
4.1. Pre-Shared Key Approach	10
4.2. Autokey	11
4.3. Network Time Security	11
4.4. External Security Protocols	11
5. NTP Security Best Practices	11
5.1. Minimizing Information Leakage	11
5.2. Avoiding Daemon Restart Attacks	12
5.3. Detection of Attacks Through Monitoring	14
5.4. Kiss-o'-Death Packets	14
5.5. Broadcast Mode Should Only Be Used On Trusted Networks	15
5.6. Symmetric Mode Should Only Be Used With Trusted Peers	15
6. NTP in Embedded Devices	15
6.1. Updating Embedded Devices	16
6.2. Server configuration	16
7. NTP over Anycast	16
8. Acknowledgments	18
9. IANA Considerations	18
10. Security Considerations	18
11. References	18
11.1. Normative References	18
11.2. Informative References	19
11.3. URIs	21
Appendix A. Best Practices specific to the Network Time Foundation implementation	21
A.1. Use enough time sources	22
A.2. NTP Control and Facility Messages	22

A.3. Monitoring	23
A.4. Leap Second File	23
A.5. Leap Smearing	23
A.6. Configuring ntpd	24
A.7. Pre-Shared Keys	24
Authors' Addresses	24

1. Introduction

NTP version 4 (NTPv4) has been widely used since its publication as [RFC5905]. This document is a collection of best practices for the operation of NTP clients and servers.

The recommendations in this document are intended to help operators distribute time on their networks more accurately and more securely. It is intended to apply generally to a broad range of networks. Some specific networks may have higher accuracy requirements that require additional techniques beyond what is documented here.

Among the best practices covered are recommendations for general network security, time protocol specific security, and NTP server and client configuration. NTP operation in embedded devices is also covered.

This document also contains information for protocol implementors who want to develop their own implementations that are compliant to 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. General Network Security Best Practices

2.1. BCP 38

Many network attacks rely on modifying the IP source address of a packet to point to a different IP address than the computer which originated it. UDP-based protocols such as NTP are generally more susceptible to spoofing attacks than connection-oriented protocols. NTP control messages can generate a lot of data in response to a small query, which makes it attractive as a vector for distributed denial-of-service attacks. (NTP Control messages are discussed further in Section 3.4). One documented instance of such an attack

can be found here [1], and further discussion in [IMC14] and [NDSS14].

BCP 38 [RFC2827] was published in 2000 to provide some level of remediation against address-spoofing attacks. BCP 38 calls for filtering outgoing and incoming traffic to make sure that the source and destination IP addresses are consistent with the expected flow of traffic on each network interface. It is RECOMMENDED that ISP's and large corporate networks implement ingress and egress filtering. More information is available at the BCP38 Info Web page [2] .

3. NTP Configuration Best Practices

This section provides Best Practices for NTP configuration and operation. Application of these best practices that are specific to the Network Time Foundation implementation, including example configuration directives valid at the time of this writing, are compiled in Appendix A.

3.1. Keeping NTP up to date

There are multiple versions of the NTP protocol in use, and multiple implementations, on many different platforms. The practices in this document are meant to apply generally to any implementation of [RFC5905]. NTP users should select an implementation that is actively maintained. Users should keep up to date on any known attacks on their selected implementation, and deploy updates containing security fixes as soon as practical.

3.2. Use enough time sources

An NTP implementation that is compliant with [RFC5905] takes the available sources of time and submits this timing data to sophisticated intersection, clustering, and combining algorithms to get the best estimate of the correct time. The description of these algorithms is beyond the scope of this document. Interested readers should read [RFC5905] or the detailed description of NTP in [MILLS2006].

- o If there is only 1 source of time, the answer is obvious. It may not be a good source of time, but it's the only source of time that can be considered. Any issue with the time at the source will be passed on to the client.
- o If there are 2 sources of time and they agree well enough, then the best time can be calculated easily. But if one source fails, then the solution degrades to the single-source solution outlined above. And if the two sources don't agree, it will be difficult

to know which one is correct without making use of information from outside of the protocol.

- o If there are 3 sources of time, there is more data available to converge on the best calculated time, and this time is more likely to be accurate. And the loss of one of the sources (by becoming unreachable or unusable) can be tolerated. But at that point, the solution degrades to the 2 source solution.
- o 4 or more sources of time is better, as long as the sources are diverse (Section 3.3). If one of these sources develops a problem there are still at least 3 other time sources.

This analysis assumes that a majority of the servers used in the solution are honest, even if some may be inaccurate. Operators should be aware of the possibility that if an attacker is in control of the network, the time coming from all servers could be compromised.

Operators who are concerned with maintaining accurate time SHOULD use at least 4 independent, diverse sources of time. Four sources will provide sufficient backup in case one source goes down. If four sources are not available, operators MAY use fewer sources, subject to the risks outlined above.

But even with 4 or more sources of time, systemic problems can happen. One example involves the leap smearing concept detailed in Section 3.7.1. For several hours before and after the June 2015 leap second, several operators configured their NTP servers with leap smearing while others did not. Many NTP end nodes could not determine an accurate time source because 2 of their 4 sources of time gave them consistent UTC/POSIX time, while the other 2 gave them consistent leap-smear time. This is just one of many potential causes of disagreement among time sources.

Operators are advised to monitor all time sources that are in use. If time sources do not generally agree, operators are encouraged to investigate the cause of this and either correct the problems or stop using defective servers. See Section 3.5 for more information.

3.3. Use a diversity of Reference Clocks

When using servers with attached hardware reference clocks, it is suggested that different types of reference clocks be used. Having a diversity of sources with independent implementations means that any one issue is less likely to cause a service interruption.

Are all clocks on a network from the same vendor? They may have the same bugs. Even devices from different vendors may not be truly independent if they share common elements. Are they using the same base chipset? Are they all running the same version of firmware? Chipset and firmware bugs can happen, but they can be more difficult to diagnose than application software bugs. When having the correct time is of critical importance, it's ultimately up to operators to ensure that their sources are sufficiently independent, even if they are not under the operator's control.

A systemic problem with time from any satellite navigation service is possible and has happened. Sunspot activity can render satellite or radio-based time source unusable. Depending on the application requirements, operators may need to consider backup scenarios in the rare circumstance when the satellite system is faulty or unavailable.

3.4. Control Messages

Some implementations of NTPv4 provide the NTP Control Messages (also known as Mode 6 messages) that were originally specified in Appendix B of [RFC1305] which defined NTPv3. These messages were never included in the NTPv4 specification, but they are still used. At the time of this writing, work is being done to formally document the structure of these control messages in [I-D.ietf-ntp-mode-6-cmds].

The NTP Control Messages are designed to permit monitoring and optionally authenticated control of NTP and its configuration. Used properly, these facilities provide vital debugging and performance information and control. But these facilities can be a vector for amplification attacks when abused. For this reason, it is RECOMMENDED that publicly-facing NTP servers should block NTP Control Message queries from outside their organization.

The ability to use NTP Control Messages beyond their basic monitoring capabilities SHOULD be limited to authenticated sessions that provide a 'controlkey'. It can also be limited through mechanisms outside of the NTP specification, such as Access Control Lists, that only allow access from approved IP addresses.

The NTP Control Messages responses are much larger than the corresponding queries. Thus, they can be abused in high-bandwidth DDoS attacks. Section 2.1 gives more information on how to provide protection for this abuse by implementing BCP 38.

3.5. Monitoring

Operators SHOULD use their NTP implementation's remote monitoring capabilities to quickly identify servers which are out of sync, and ensure correctness of the service. Operators SHOULD also monitor system logs for messages so problems and abuse attempts can be quickly identified.

If a system starts to receive NTP Reply packets from a remote time server that do not correspond to any requests sent by the system, that can be an indication that an attacker is forging that system's IP address in requests to the remote time server. The goal of this attack is to adversely impact the availability of time to the targeted system whose address is being forged. Based on these forged packets, the remote time server might decide to throttle or rate limit packets, or even stop sending packets to the targeted system.

If a system is a broadcast client and its system log shows that it is receiving early time messages from its server, that is an indication that somebody may be forging packets from a broadcast server. (Broadcast client and server modes are defined in Section 3 of [RFC5905])

If a server's system log shows messages that indicates it is receiving NTP timestamps that are much earlier than the current system time, then either the system clock is unusually fast or somebody is trying to launch a replay attack against that server.

3.6. Using Pool Servers

It only takes a small amount of bandwidth and system resources to synchronize one NTP client, but NTP servers that can service tens of thousands of clients take more resources to run. Network operators and advanced users who want to synchronize their computers MUST only synchronize to servers that they have permission to use.

The NTP Pool Project is a group of volunteers who have donated their computing and bandwidth resources to freely distribute time from primary time sources to others on the Internet. The time is generally of good quality but comes with no guarantee whatsoever. If you are interested in using this pool, please review their instructions at <http://www.pool.ntp.org/en/use.html> [3].

Vendors can obtain their own subdomain that is part of the NTP Pool Project. This offers vendors the ability to safely make use of the time distributed by the pool for their devices. Details are available at <http://www.pool.ntp.org/en/vendors.html> [4] .

If there is a need to synchronize many computers, an operator may want to run local NTP servers that are synchronized to the NTP Pool Project. NTP users on that operator's networks can then be synchronized to local NTP servers.

3.7. Leap Second Handling

UTC is kept in agreement with the astronomical time UT1 [5] to within ± 0.9 seconds by the insertion (or possibly a deletion) of a leap second. UTC is an atomic time scale whereas UT1 is based on the rotational rate of the earth. Leap seconds are not introduced at a fixed rate. They are announced by the International Earth Rotation and Reference Systems Service (IERS) in its Bulletin C [6] when necessary to keep UTC and UT1 aligned.

NTP time is based on the UTC timescale, and the protocol has the capability to broadcast leap second information. Some Global Navigation Satellite Systems (like GPS) or radio transmitters (like DCF77) broadcast leap second information. If an NTP client is synced to an NTP server that provides leap second notification, the client will get advance notification of impending leap seconds automatically.

Since the length of the UT1 day is generally slowly increasing [7], all leap seconds that have been introduced since the practice started in 1972 have been positive leap seconds, where a second is added to UTC. NTP also supports a negative leap second, where a second is removed from UTC, if that ever becomes necessary.

While earlier versions of NTP contained some ambiguity regarding when a leap second that is broadcast by a server should be applied by a client, RFC 5905 is clear that leap seconds are only applied on the last day of a month. However, because some older clients may apply it at the end of the current day, it is RECOMMENDED that NTP servers wait until the last day of the month before broadcasting leap seconds. Doing this will prevent older clients from applying a leap second at the wrong time. When implementing this recommendation, operators should ensure that clients are not configured to use polling intervals greater than 24 hours, so the leap second notification is not missed.

In circumstances where an NTP server is not receiving leap second information from an automated source, certain organizations maintain files which are updated every time a new leap second is announced:

NIST: <ftp://time.nist.gov/pub/leap-seconds.list>

US Navy (maintains GPS Time): <ftp://tycho.usno.navy.mil/pub/ntp/leap-seconds.list>

IERS (announces leap seconds):
<https://hpiers.obspm.fr/iers/bul/bulc/ntp/leap-seconds.list>

3.7.1. Leap Smearing

Some NTP installations make use of a technique called Leap Smearing. With this method, instead of introducing an extra second (or eliminating a second) on a leap second event, NTP time will be slewed in small increments over a comparably large window of time (called the smear interval) around the leap second event. The smear interval should be large enough to make the rate that the time is slewed small, so that clients will follow the smeared time without objecting. Periods ranging from 2 to 24 hours have been used successfully. During the adjustment window, all the NTP clients' times may be offset from UTC by as much as a full second, depending on the implementation. But at least all clients will generally agree on what time they think it is.

The purpose of Leap Smearing is to enable systems that don't deal with the leap second event properly to function consistently, at the expense of fidelity to UTC during the smear window. During a standard leap second event, that minute will have 61 (or possibly 59) seconds in it, and some applications (and even some OS's) are known to have problems with that.

Operators who have legal obligations or other strong requirements to be synchronized with UTC or civil time SHOULD NOT use leap smearing, because the distributed time cannot be guaranteed to be traceable to UTC during the smear interval.

Clients that are connected to leap smearing servers MUST NOT apply the standard NTP leap second handling. These clients must never have a leap second file loaded, and the smearing servers must never advertise to clients that a leap second is pending.

Any use of leap smearing servers should be limited to within a single, well-controlled environment. Leap Smearing MUST NOT be used for public-facing NTP servers, as they will disagree with non-smearing servers (as well as UTC) during the leap smear interval, and there is no standardized way for a client to detect that a server is using leap smearing. However, be aware that some public-facing servers may be configured this way anyway in spite of this guidance.

System Administrators are advised to be aware of impending leap seconds and how the servers (inside and outside their organization)

they are using deal with them. Individual clients MUST NOT be configured to use a mixture of smeared and non-smeared servers. If a client uses smeared servers, the servers it uses must all have the same leap smear configuration.

4. NTP Security Mechanisms

In the standard configuration NTP packets are exchanged unprotected between client and server. An adversary that is able to become a Man-In-The-Middle is therefore able to drop, replay or modify the content of the NTP packet, which leads to degradation of the time synchronization or the transmission of false time information. A threat analysis for time synchronization protocols is given in [RFC7384]. NTP provides two internal security mechanisms to protect authenticity and integrity of the NTP packets. Both measures protect the NTP packet by means of a Message Authentication Code (MAC). Neither of them encrypts the NTP's payload, because this payload information is not considered to be confidential.

4.1. Pre-Shared Key Approach

This approach applies a symmetric key for the calculation of the MAC, which protects authenticity and integrity of the exchanged packets for an association. NTP does not provide a mechanism for the exchange of the keys between the associated nodes. Therefore, for each association, keys MUST be exchanged securely by external means, and they MUST be protected from disclosure. It is RECOMMENDED that each association be protected by its own unique key. It is RECOMMENDED that participants agree to refresh keys periodically. However, NTP does not provide a mechanism to assist in doing so. Each communication partner will need to keep track of its keys in its own local key storage.

[RFC5905] specifies using the MD5 hash algorithm for calculation of the MAC, but other algorithms may be supported as well. The MD5 hash is now considered to be too weak and unsuitable for cryptographic usage. [RFC6151] has more information on the algorithm's weaknesses. Implementations will soon be available based on AES-128-CMAC [I-D.ietf-ntp-mac], and users SHOULD use that when it is available.

Some implementations store the key in clear text. Therefore it MUST only be readable by the NTP process.

An NTP client has to be able to link a key to a particular server in order to establish a protected association. This linkage is implementation specific. Once applied, a key will be trusted until the link is removed.

4.2. Autokey

[RFC5906] specifies the Autokey protocol. It was published in 2010 to provide automated key management and authentication of NTP servers. However, security researchers have identified vulnerabilities [8] in the Autokey protocol.

Autokey SHOULD NOT be used.

4.3. Network Time Security

Work is in progress on an enhanced replacement for Autokey. Refer to [I-D.ietf-ntp-using-nts-for-ntp] for more information.

4.4. External Security Protocols

If applicable, external security protocols such as IPsec and MACsec can be applied to enhance integrity and authenticity protection of NTP time synchronization packets. Usage of such external security protocols can decrease time synchronization performance [RFC7384]. Therefore, operators are advised to carefully evaluate if the decreased time synchronization performance meets their specific timing requirements.

Note that none of the security measures described in Section 4 can prevent packet delay manipulation attacks on NTP. Such delay attacks can target time synchronization packets sent as clear-text or even within an encrypted tunnel. These attacks are described further in Section 3.2.6 of [RFC7384].

5. NTP Security Best Practices

This section lists some general NTP security practices, but these issues may (or may not) have been mitigated in particular versions of particular implementations. Contact the maintainers of the relevant implementation for more information.

5.1. Minimizing Information Leakage

The base NTP packet leaks important information (including reference ID and reference time) that may be used in attacks [NDSS16], [CVE-2015-8138], [CVE-2016-1548]. A remote attacker can learn this information by sending mode 3 queries to a target system and inspecting the fields in the mode 4 response packet. NTP control queries also leak important information (including reference ID, expected origin timestamp, etc.) that may be used in attacks [CVE-2015-8139]. A remote attacker can learn this information by

sending control queries to a target system and inspecting the leaked information in the response.

As such, mechanisms outside of the NTP protocol, such as Access Control Lists, SHOULD be used to limit the exposure of this information to allowed IP addresses, and keep it from remote attackers not on the list. Hosts SHOULD only respond to NTP control queries from authorized parties.

An NTP client that does not provide time on the network can additionally log and drop incoming mode 3 timing queries from unexpected sources. Note well that the easiest way to monitor the status of an NTP instance is to send it a mode 3 query, so it may not be desirable to drop all mode 3 queries. As an alternative, operators SHOULD either filter mode 3 queries from outside their networks, or make sure mode 3 queries are allowed only from trusted systems or networks.

A "leaf-node host" is a host that is using NTP solely for the purpose of adjusting its own system time. Such a host is not expected to provide time to other hosts, and relies exclusively on NTP's basic mode to take time from a set of servers. (That is, the host sends mode 3 queries to its servers and receives mode 4 responses from these servers containing timing information.) To minimize information leakage, leaf-node hosts SHOULD drop all incoming NTP packets except mode 4 response packets that come from known sources. An exception to this can be made if a leaf-node host is being actively monitored, in which case incoming packets from the monitoring server can be allowed.

Please refer to [I-D.ietf-ntp-data-minimization] for more information.

5.2. Avoiding Daemon Restart Attacks

[RFC5905] says NTP clients should not accept time shifts greater than the panic threshold. Specifically, RFC 5905 says "PANIC means the offset is greater than the panic threshold PANICT (1000 s) and SHOULD cause the program to exit with a diagnostic message to the system log."

However, this behavior can be exploited by attackers as described in [NDSS16], when the following two conditions hold:

1. The operating system automatically restarts the NTP client when it quits. (Modern *NIX operating systems are replacing traditional init systems with process supervisors, such as systemd, which can be configured to automatically restart any

daemons that quit. This behavior is the default in CoreOS and Arch Linux. As of the time of this writing, it appears likely to become the default behavior in other systems as they migrate legacy init scripts to process supervisors such as systemd.)

2. The NTP client is configured to ignore the panic threshold on all restarts.

In such cases, if the attacker can send the target an offset that exceeds the panic threshold, the client will quit. Then, when it restarts, it ignores the panic threshold and accepts the attacker's large offset.

Operators need to be aware that when operating with the above two conditions, the panic threshold offers no protection from attacks. The natural solution is not to run hosts with these conditions. Specifically, operators SHOULD NOT ignore the panic threshold in all cold-start situations unless sufficient oversight and checking is in place to make sure that this type of attack cannot happen.

As an alternative, the following steps MAY be taken by operators to mitigate the risk of attack:

- o Monitor the NTP system log to detect when the NTP daemon has quit due to a panic event, as this could be a sign of an attack.
- o Request manual intervention when a timestep larger than the panic threshold is detected.
- o Configure the ntp client to only ignore the panic threshold in a cold start situation.
- o Increase the minimum number of servers required before the NTP client adjusts the system clock. This will make the NTP client wait until enough trusted sources of time agree before declaring the time to be correct.

In addition, the following steps SHOULD be taken by those who implement the NTP protocol:

- o Prevent the NTP daemon from taking time steps that set the clock to a time earlier than the compile date of the NTP daemon.
- o Prevent the NTP daemon from putting 'INIT' in the reference ID of its NTP packets upon initializing. This will make it more difficult for attackers to know when the daemon reboots.

5.3. Detection of Attacks Through Monitoring

Operators SHOULD monitor their NTP instances to detect attacks. Many known attacks on NTP have particular signatures. Common attack signatures include:

1. Bogus packets - A packet whose origin timestamp does not match the value that expected by the client.
2. Zero origin packet - A packet with an origin timestamp set to zero [CVE-2015-8138].
3. A packet with an invalid cryptographic MAC [CCR16].

The observation of many such packets could indicate that the client is under attack.

5.4. Kiss-o'-Death Packets

The "Kiss-o'-Death" (KoD) packet includes a rate management mechanism where a server can tell a misbehaving client to reduce its query rate. KoD packets in general (and the RATE packet in particular) are defined in Section 7.4 of [RFC5905]. It is RECOMMENDED that all NTP devices respect these packets and back off when asked to do so by a server. It is even more important for an embedded device, which may not have an exposed control interface for NTP.

That said, a client MUST only accept a KoD packet if it has a valid origin timestamp. Once a RATE packet is accepted, the client should increase its poll interval value (thus decreasing its polling rate) up to a reasonable maximum. This maximum can vary by implementation but should not exceed a poll interval value of 13 (2 hours). The mechanism to determine how much to increase the poll interval value is undefined in [RFC5905]. If the client uses the poll interval value sent by the server in the RATE packet, it MUST NOT simply accept any value. Using large interval values may open a vector for a denial-of-service attack that causes the client to stop querying its server [NDSS16].

The KoD rate management mechanism relies on clients behaving properly in order to be effective. Some clients ignore the RATE packet entirely, and other poorly-implemented clients might unintentionally increase their poll rate and simulate a denial of service attack. Server administrators are advised to be prepared for this and take measures outside of the NTP protocol to drop packets from misbehaving clients when these clients are detected.

Kiss-o'-Death (KoD) packets can be used in denial of service attacks. Thus, the observation of even just one RATE packet with a high poll value could be sign that the client is under attack. And KoD packets are commonly accepted even when not cryptographically authenticated, which increases the risk of denial of service attacks.

5.5. Broadcast Mode Should Only Be Used On Trusted Networks

Per [RFC5905], NTP's broadcast mode is authenticated using symmetric key cryptography. The broadcast server and all its broadcast clients share a symmetric cryptographic key, and the broadcast server uses this key to append a message authentication code (MAC) to the broadcast packets it sends.

Importantly, all broadcast clients that listen to this server have to know the cryptographic key. This mean that any client can use this key to send valid broadcast messages that look like they come from the broadcast server. Thus, a rogue broadcast client can use its knowledge of this key to attack the other broadcast clients.

For this reason, an NTP broadcast server and all its clients have to trust each other. Broadcast mode SHOULD only be run from within a trusted network.

5.6. Symmetric Mode Should Only Be Used With Trusted Peers

In symmetric mode, two peers Alice and Bob can both push and pull synchronization to and from each other using either ephemeral symmetric passive (mode 2) or persistent symmetric active (NTP mode 1) packets. The persistent association is preconfigured and initiated at the active peer but not preconfigured at the passive peer (Bob). Upon receipt of a mode 1 NTP packet from Alice, Bob mobilizes a new ephemeral association if he does not have one already. This is a security risk for Bob because an arbitrary attacker can attempt to change Bob's time by asking Bob to become its symmetric passive peer.

For this reason, a host SHOULD only allow symmetric passive associations to be established with trusted peers. Specifically, a host SHOULD require each of its symmetric passive association to be cryptographically authenticated. Each symmetric passive association SHOULD be authenticated under a different cryptographic key.

6. NTP in Embedded Devices

As computing becomes more ubiquitous, there will be many small embedded devices that require accurate time. These devices may not have a persistent battery-backed clock, so using NTP to set the

correct time on power-up may be critical for proper operation. These devices may not have a traditional user interface, but if they connect to the Internet they will be subject to the same security threats as traditional deployments.

6.1. Updating Embedded Devices

Vendors of embedded devices are advised to pay attention to the current state of protocol security issues and bugs in their chosen implementation, because their customers don't have the ability to update their NTP implementation on their own. Those devices may have a single firmware upgrade, provided by the manufacturer, that updates all capabilities at once. This means that the vendor assumes the responsibility of making sure their devices have an up-to-date and secure NTP implementation.

Vendors of embedded devices SHOULD include the ability to update the list of NTP servers used by the device.

There is a catalog of NTP server abuse incidents, some of which involve embedded devices, on the Wikipedia page for NTP Server Misuse and Abuse [9].

6.2. Server configuration

Vendors of embedded devices with preconfigured NTP servers need to carefully consider which servers to use. There are several public-facing NTP servers available, but they may not be prepared to service requests from thousands of new devices on the Internet. Vendors MUST only preconfigure NTP servers that they have permission to use.

Vendors are encouraged to invest resources into providing their own time servers for their devices to connect to. This may be done through the NTP Pool Project, as documented in Section 3.6.

Vendors should read [RFC4085], which advises against embedding globally-routable IP addresses in products, and offers several better alternatives.

7. NTP over Anycast

Anycast is described in BCP 126 [RFC4786]. (Also see [RFC7094]). With anycast, a single IP address is assigned to multiple servers, and routers direct packets to the closest active server.

Anycast is often used for Internet services at known IP addresses, such as DNS. Anycast can also be used in large organizations to simplify configuration of many NTP clients. Each client can be

configured with the same NTP server IP address, and a pool of anycast servers can be deployed to service those requests. New servers can be added to or taken from the pool, and other than a temporary loss of service while a server is taken down, these additions can be transparent to the clients.

Note well that using a single anycast address for NTP presents its own potential issues. It means each client will likely use a single time server source. A key element of a robust NTP deployment is each client using multiple sources of time. With multiple time sources, a client will analyze the various time sources, selecting good ones, and disregarding poor ones. If a single Anycast address is used, this analysis will not happen. This can be mitigated by creating multiple, separate anycast pools so clients can have multiple sources of time while still gaining the configuration benefits of the anycast pools.

If clients are connected to an NTP server via anycast, the client does not know which particular server they are connected to. As anycast servers enter and leave the network, or the network topology changes, the server a particular client is connected to may change. This may cause a small shift in time from the perspective of the client when the server it is connected to changes. In extreme cases where the network topology is changing rapidly, this could cause the server seen by a client to rapidly change as well, which can lead to larger time inaccuracies. It is RECOMMENDED that network operators only deploy anycast NTP in environments where operators know these small shifts can be tolerated by the applications running on the clients being synchronized in this manner.

Configuration of an anycast interface is independent of NTP. Clients will always connect to the closest server, even if that server is having NTP issues. It is RECOMMENDED that anycast NTP implementations have an independent method of monitoring the performance of NTP on a server. If the server is not performing to specification, it should remove itself from the Anycast network. It is also RECOMMENDED that each Anycast NTP server have an alternative method of access, such as an alternate Unicast IP address, so its performance can be checked independently of the anycast routing scheme.

One useful application in large networks is to use a hybrid unicast/anycast approach. Stratum 1 NTP servers can be deployed with unicast interfaces at several sites. Each site may have several Stratum 2 servers with two ethernet interfaces, or a single interface which can support multiple addresses. One interface has a unique unicast IP address. The second has an anycast IP interface (with a shared IP address per location). The unicast interfaces can be used to obtain

time from the Stratum 1 servers globally (and perhaps peer with the other Stratum 2 servers at their site). Clients at each site can be configured to use the shared anycast address for their site, simplifying their configuration. Keeping the anycast routing restricted on a per-site basis will minimize the disruption at the client if its closest anycast server changes. Each Stratum 2 server can be uniquely identified on their unicast interface, to make monitoring easier.

8. Acknowledgments

The authors wish to acknowledge the contributions of Sue Graves, Samuel Weiler, Lisa Perdue, Karen O'Donoghue, David Malone, Sharon Goldberg, Martin Burnicki, Miroslav Lichvar, Daniel Fox Franke, Robert Nagy, and Brian Haberman.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

Time is a fundamental component of security on the internet. The absence of a reliable source of current time subverts many common web authentication schemes, e.g., by allowing the use of expired credentials or by allowing for replay of messages only intended to be processed once.

Much of this document directly addresses how to secure NTP servers. In particular, see Section 2, Section 4, and Section 5.

There are several general threats to time synchronization protocols which are discussed in [RFC7384].

[I-D.ietf-ntp-using-nts-for-ntp] specifies the Network Time Security (NTS) mechanism and applies it to NTP. Readers are encouraged to check the status of the draft, and make use of the methods it describes.

11. References

11.1. 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>>.

- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC4085] Plonka, D., "Embedding Globally-Routable Internet Addresses Considered Harmful", BCP 105, RFC 4085, DOI 10.17487/RFC4085, June 2005, <<https://www.rfc-editor.org/info/rfc4085>>.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <<https://www.rfc-editor.org/info/rfc4786>>.
- [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>>.

11.2. Informative References

- [CCR16] Malhotra, A. and S. Goldberg, "Attacking NTP's Authenticated Broadcast Mode", SIGCOMM Computer Communications Review (CCR) , 2016.
- [CVE-2015-8138] Van Gundy, M. and J. Gardner, "NETWORK TIME PROTOCOL ORIGIN TIMESTAMP CHECK IMPERSONATION VULNERABILITY", 2016, <<http://www.talosintel.com/reports/TALOS-2016-0077>>.
- [CVE-2015-8139] Van Gundy, M., "NETWORK TIME PROTOCOL NTPQ AND NTPDC ORIGIN TIMESTAMP DISCLOSURE VULNERABILITY", 2016, <<http://www.talosintel.com/reports/TALOS-2016-0078>>.
- [CVE-2016-1548] Gardner, J. and M. Lichvar, "Xleave Pivot: NTP Basic Mode to Interleaved", 2016, <http://blog.talosintel.com/2016/04/vulnerability-spotlight-further-ntpd_27.html>.

- [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.ietf-ntp-mac]
Malhotra, A. and S. Goldberg, "Message Authentication Code for the Network Time Protocol", draft-ietf-ntp-mac-06 (work in progress), January 2019.
- [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-06 (work in progress), September 2018.
- [I-D.ietf-ntp-using-nts-for-ntp]
Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", draft-ietf-ntp-using-nts-for-ntp-17 (work in progress), February 2019.
- [IMC14] Czyz, J., Kallitsis, M., Gharaibeh, M., Papadopoulos, C., Bailey, M., and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks", Internet Measurement Conference , 2014.
- [MILLS2006]
Mills, D., "Computer network time synchronization: the Network Time Protocol", CRC Press , 2006.
- [NDSS14] Rossow, C., "Amplification Hell: Revisiting Network Protocols for DDoS Abuse", NDSS'14, San Diego, CA. , 2014.
- [NDSS16] Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", NDSS'16, San Diego, CA. , 2016, <<https://eprint.iacr.org/2015/1020.pdf>>.
- [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>>.
- [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>>.

- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.
- [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <<https://www.rfc-editor.org/info/rfc7094>>.
- [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>>.

11.3. URIs

- [1] <https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/>
- [2] <http://www.bcp38.info>
- [3] <http://www.pool.ntp.org/en/use.html>
- [4] <http://www.pool.ntp.org/en/vendors.html>
- [5] https://en.wikipedia.org/wiki/Solar_time#Mean_solar_time
- [6] <https://www.iers.org/IERS/EN/Publications/Bulletins/bulletins.html>
- [7] https://en.wikipedia.org/wiki/Solar_time#Mean_solar_time
- [8] <https://lists.ntp.org/pipermail/ntpwg/2011-August/001714.html>
- [9] https://en.wikipedia.org/wiki/NTP_server_misuse_and_abuse
- [10] <http://www.ntp.org/downloads.html>
- [11] <http://bk1.ntp.org/ntp-stable/README.leapsmear?PAGE=anno>
- [12] <https://support.ntp.org/bin/view/Support/ConfiguringNTP>

Appendix A. Best Practices specific to the Network Time Foundation implementation

The Network Time Foundation (NTF) provides a widely used implementation of NTP, known as ntpd [10]. It is an evolution of the first NTP implementations developed by David Mills at the University

of Delaware. This appendix contains additional recommendations specific to this implementation that are valid at the time of this writing.

A.1. Use enough time sources

In addition to the recommendation given in Section 3.2 the ntpd implementation provides the 'pool' directive. Starting with ntp-4.2.6, using this directive in the ntp.conf file will spin up enough associations to provide robust time service, and will disconnect poor servers and add in new servers as-needed. The 'minclock' and 'maxclock' options of the 'tos' command may be used to override the default values of how many servers are discovered through the 'pool' directive.

A.2. NTP Control and Facility Messages

In addition to NTP Control Messages the ntpd implementation also offers the Mode 7 commands for monitoring and configuration.

If Mode 7 has been explicitly enabled to be used for more than basic monitoring it should be limited to authenticated sessions that provide a 'requestkey'.

As mentioned above, there are two general ways to use Mode 6 and Mode 7 requests. One way is to query ntpd for information, and this mode can be disabled with:

```
restrict ... noquery
```

The second way to use Mode 6 and Mode 7 requests is to modify ntpd's behavior. Modification of ntpd's configuration requires an authenticated session by default. If no authentication keys have been specified no modifications can be made. For additional protection, the ability to perform these modifications can be controlled with:

```
restrict ... nomodify
```

Users can prevent their NTP servers from considering query/configuration traffic by default by adding the following to their ntp.conf file:

```
restrict default -4 nomodify notrap nopeer noquery
```

```
restrict default -6 nomodify notrap nopeer noquery
```

```
restrict source nomodify notrap noquery
```

A.3. Monitoring

The ntpd implementation allows remote monitoring. Access to this service is generally controlled by the "noquery" directive in NTP's configuration file (ntp.conf) via a "restrict" statement. The syntax reads:

```
restrict address mask address_mask noquery
```

If a system is using broadcast mode and is running ntp-4.2.8p6 or later, use the 4th field of the ntp.keys file to specify the IPs of machines that are allowed to serve time to the group.

A.4. Leap Second File

The use of leap second files requires ntpd 4.2.6 or later. After fetching the leap seconds file onto the server, add this line to ntpd.conf to apply and use the file, substituting the proper path:

```
leapfile "/path/to/leap-file"
```

There may need to restart ntpd to apply this change.

ntpd servers with a manually configured leap second file will ignore leap second information broadcast from upstream NTP servers until the leap second file expires. If no valid leap second file is available then a leap second notification from an attached reference clock is always accepted by ntpd.

If no valid leap second file is available, a leap second notification may be accepted from upstream NTP servers. As of ntp-4.2.6, a majority of servers must provide the notification before it is accepted. Before 4.2.6, a leap second notification would be accepted if a single upstream server or a group of configured servers provided a leap second notification. This would lead to misbehavior if single NTP servers sent an invalid leap second warning, e.g. due to a faulty GPS receiver in one server, but this behavior was once chosen because in the "early days" there was a greater chance that leap second information would be available from a very limited number of sources.

A.5. Leap Smearing

Leap Smearing was introduced in ntpd versions 4.2.8.p3 and 4.3.47, in response to client requests. Support for leap smearing is not configured by default and must be added at compile time. In addition, no leap smearing will occur unless a leap smear interval is specified in ntpd.conf. For more information, refer to <http://bk.ntp.org/ntp-stable/README.leapsmear?PAGE=anno> [11].

A.6. Configuring ntpd

See <https://support.ntp.org/bin/view/Support/ConfiguringNTP> [12] for additional information on configuring ntpd.

A.7. Pre-Shared Keys

Each communication partner must add the key information to their key file in the form:

```
keyid type key
```

where "keyid" is a number between 1 and 65534, inclusive, "type" is an ASCII character which defines the key format, and "key" is the key itself.

An ntpd client establishes a protected association by appending the option "key keyid" to the server statement in ntp.conf:

```
server address key keyid
```

substituting the server address in the "address" field and the numerical keyid to use with that server in the "keyid" field.

A key is deemed trusted when its keyid is added to the list of trusted keys by the "trustedkey" statement in ntp.conf.

```
trustedkey keyid_1 keyid_2 ... keyid_n
```

Starting with ntp-4.2.8p7 the ntp.keys file accepts an optional 4th column, a comma-separated list of IPs that are allowed to serve time. Use this feature. Note, however, that an adversarial client that knows the symmetric broadcast key could still easily spoof its source IP to an IP that is allowed to serve time. (This is easy to do because the origin timestamp on broadcast mode packets is not validated by the client. By contrast, client/server and symmetric modes do require origin timestamp validation, making it more difficult to spoof packets [CCR16]).

Authors' Addresses

Denis Reilly (editor)
Orolia USA
1565 Jefferson Road, Suite 460
Rochester, NY 14623
US

Email: denis.reilly@orolia.com

Harlan Stenn
Network Time Foundation
P.O. Box 918
Talent, OR 97540
US

Email: stenn@nwttime.org

Dieter Sibold
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8420
Fax: +49-531-592-698420
Email: dieter.sibold@ptb.de

Network Working Group
Internet-Draft
Updates: 5905 (if approved)
Intended status: Standards Track
Expires: September 26, 2019

D. Franke
Akamai
A. Malhotra
Boston University
March 25, 2019

NTP Client Data Minimization
draft-ietf-ntp-data-minimization-04

Abstract

This memo proposes backward-compatible updates to the Network Time Protocol to strip unnecessary identifying information from client requests and to improve resilience against blind spoofing of unauthenticated server responses.

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 26, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Requirements Language	2
3. Client Packet Format	2
4. Security and Privacy Considerations	3
4.1. Data Minimization	3
4.2. Transmit Timestamp Randomization	4
5. IANA Considerations	4
6. Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION	4
7. References	5
7.1. Normative References	5
7.2. Informative References	5
Appendix A. Acknowledgements	6
Authors' Addresses	6

1. Introduction

Network Time Protocol (NTP) packets, as specified by RFC 5905 [RFC5905], carry a great deal of information about the state of the NTP daemon which transmitted them. In the case of mode 4 packets (responses sent from server to client), as well as in broadcast (mode 5) and symmetric peering modes (mode 1/2), most of this information is essential for accurate and reliable time synchronization. However, in mode 3 packets (requests sent from client to server), most of these fields serve no purpose. Server implementations never need to inspect them, and they can achieve nothing by doing so. Populating these fields with accurate information is harmful to privacy of clients because it allows a passive observer to fingerprint clients and track them as they move across networks.

This memo updates RFC 5905 to redact unnecessary data from mode 3 packets. This is a fully backwards-compatible proposal. It calls for no changes on the server side, and clients which implement these updates will remain fully interoperable with existing servers.

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. Client Packet Format

In every client-mode packet sent by a Network Time Protocol [RFC5905] implementation:

The first octet, which contains the leap indicator, version number, and mode fields, SHOULD be set to 0x23 (LI = 0, VN = 4, Mode = 3).

The Transmit Timestamp field SHOULD be set uniformly at random, generated by a mechanism suitable for cryptographic purposes. [RFC4086] provides guidance on the generation of random values.

The Poll field SHOULD be set to either the actual polling interval as specified by RFC 5905 or zero.

The Precision field SHOULD be set to 0x20.

All other header fields, specifically the Stratum, Root Delay, Root Dispersion, Reference ID, Reference Timestamp, Origin Timestamp, and Receive Timestamp, SHOULD be set to zero.

Servers MUST allow client packets to conform to the above recommendations. This requirement shall not be construed so as to prohibit servers from rejecting conforming packets for unrelated reasons, such as access control or rate limiting.

4. Security and Privacy Considerations

4.1. Data Minimization

Zeroing out unused fields in client requests prevents disclosure of information that can be used for fingerprinting [RFC6973].

While populating any of these fields with authentic data reveals at least some identifying information about the client, the Origin Timestamp and Receive Timestamp fields constitute a particularly severe information leak. RFC 5905 calls for clients to copy the transmit timestamp and destination timestamp of the server's most recent response into the origin timestamp and receive timestamp (respectively) of their next request to that server. Therefore, when a client moves between networks, a passive observer of both network paths can determine with high confidence that the old and new IP addresses belong to the same system by noticing that the transmit timestamp of a response sent to the old IP matches the origin timestamp of a request sent from the new one.

Zeroing the poll field is made optional (MAY rather than SHOULD) so as not to preclude future development of schemes wherein the server uses information about the client's current poll interval in order to recommend adjustments back to the client. Putting accurate information into this field has no significant impact on privacy

since an observer can already obtain this information simply by observing the actual interval between requests.

4.2. Transmit Timestamp Randomization

While this memo calls for most fields in client packets to be set to zero, the transmit timestamp SHOULD be randomized. This decision is motivated by security as well as privacy.

NTP servers copy the transmit timestamp from the client's request into the origin timestamp of the response; this memo calls for no change in this behavior. Clients discard any response whose origin timestamp does not match the transmit timestamp of any request currently in flight.

In the absence of cryptographic authentication, verification of origin timestamps is clients' primary defense against blind spoofing of NTP responses. It is therefore important that clients' transmit timestamps be unpredictable. Their role in this regard is closely analogous to that of TCP Initial Sequence Numbers [RFC6528].

The traditional behavior of the NTP reference implementation is to randomize only a few (typically 10-15 depending on the precision of the system clock) low-order bits of transmit timestamp, with all higher bits representing the system time, as measured just before the packet was sent. This is suboptimal, because with so few random bits, an adversary sending spoofed packets at high volume will have a good chance of correctly guessing a valid origin timestamp.

5. IANA Considerations

[RFC EDITOR: DELETE PRIOR TO PUBLICATION]

This memo introduces no new IANA considerations.

6. Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in 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.

As of today the following vendors have produced an implementation of the NTP Client Data Minimization recommendations described in this document.

OpenNTPD

7. References

7.1. 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>>.

7.2. Informative References

- [RFC2030] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 2030, DOI 10.17487/RFC2030, October 1996, <<https://www.rfc-editor.org/info/rfc2030>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC6528] Gont, F. and S. Bellovin, "Defending against Sequence Number Attacks", RFC 6528, DOI 10.17487/RFC6528, February 2012, <<https://www.rfc-editor.org/info/rfc6528>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.

7.3. URIs

[1] <https://github.com/openbsd/src/commit/1346900e6d0ac3aeb0e3f9eb60b94c66586978c6>

Appendix A. Acknowledgements

The possibility of minimizing data in client packets was described in RFC 2030 [RFC2030]. The authors would like to acknowledge Alexander Guy for pioneering the idea of randomization of all bits of the transmit timestamp in the rdate program of the OpenBSD project as early as May 2004 [1].

The authors would also like to thank Prof. Sharon Goldberg and Miroslav Lichvar for encouraging standardisation of the approach described in this document.

Authors' Addresses

Daniel Fox Franke
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02142
United States

Email: dafranke@akamai.com
URI: <https://www.dfranke.us>

Aanchal Malhotra
Boston University
111 Cummington St
Boston, MA/ 02215
United States

Email: aanchal4@bu.edu

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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 21 April 2022.

Copyright Notice

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

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

Table of Contents

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

1. Introduction

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

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

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

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

4. Physical layer (hardware - PHY chip)

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

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

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

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

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

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

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

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

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

This document assumes familiarity with RFC 5905.

1.1. Requirements Language

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

2. Interleaved Client/server mode

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

An example of packets in a client/server exchange using the interleaved mode is shown in Figure 1. The packets in the basic and interleaved mode are indicated with B and I respectively. The timestamps $t1^*$, $t3^*$ 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).

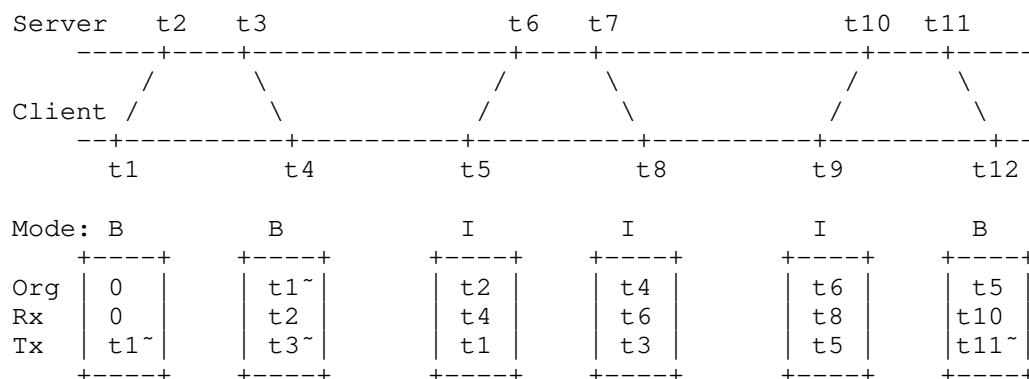


Figure 1: Packet timestamps in interleaved client/server mode

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

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

3. Interleaved Symmetric mode

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

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

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

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

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

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

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

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

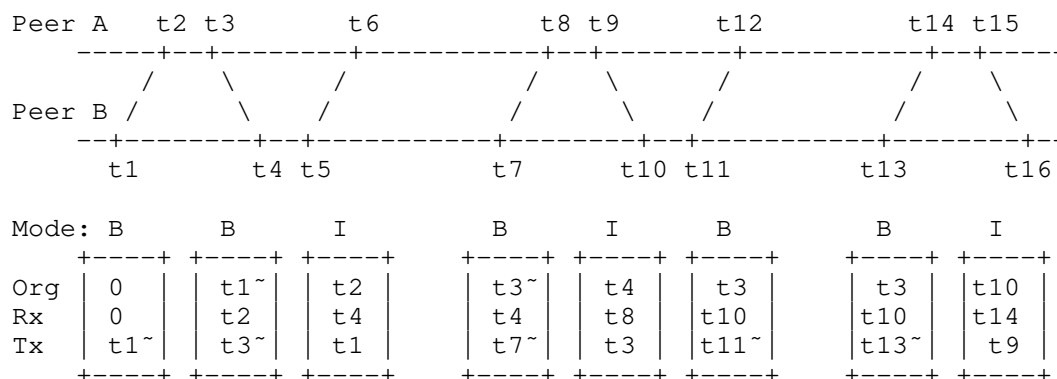


Figure 2: Packet timestamps in interleaved symmetric mode

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

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

4. Interleaved Broadcast mode

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

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

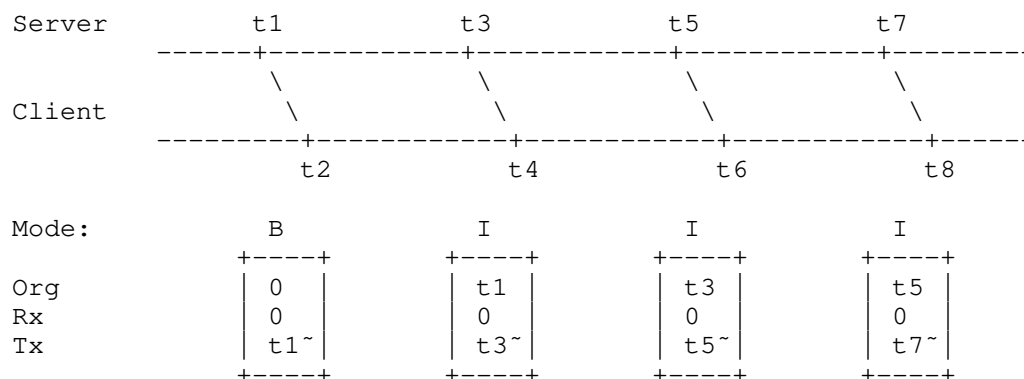


Figure 3: Packet timestamps in interleaved broadcast mode

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

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

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

5. Protocol Failures

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

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

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

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

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

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

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

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

6. Security Considerations

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

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

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

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

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

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

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

7. IANA Considerations

This memo includes no request to IANA.

8. Acknowledgements

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

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

9. References

9.1. Normative References

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

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

9.2. Informative References

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

Authors' Addresses

Miroslav Lichvar
Red Hat
Purkynova 115
612 00 Brno
Czech Republic

Email: mlichvar@redhat.com

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

Email: aanchal4@bu.edu

Internet Engineering Task Force
Internet-Draft
Updates: 5905 (if approved)
Intended status: Standards Track
Expires: July 8, 2019

A. Malhotra
S. Goldberg
Boston University
January 4, 2019

Message Authentication Code for the Network Time Protocol
draft-ietf-ntp-mac-06

Abstract

RFC 5905 states that Network Time Protocol (NTP) packets should be authenticated by appending the NTP data to a 128-bit key, and hashing the result with MD5 to obtain a 128-bit tag. This document deprecates MD5-based authentication, which is considered to be too weak, and recommends the use of AES-CMAC as in RFC 4493 as a replacement.

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 8, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	2
2. Deprecating the use of MD5	2
3. Replacement Recommendation	3
4. Motivation	3
5. Test Vectors	3
6. Security Considerations	3
7. Acknowledgements	4
8. IANA Considerations	4
9. References	4
9.1. Normative References	4
9.2. Informative References	4
Authors' Addresses	5

1. Introduction

RFC 5905 [RFC5905] states that Network Time Protocol (NTP) packets should be authenticated by appending the NTP data to a 128-bit key, and hashing the result with MD5 to obtain a 128-bit tag. This document deprecates MD5-based authentication, which is considered to be too weak, and recommends the use of AES-CMAC [RFC4493] as a replacement.

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. Deprecating the use of MD5

RFC 5905 [RFC5905] defines how the MD5 digest algorithm in RFC 1321 [RFC1321] can be used as a message authentication code (MAC) for authenticating NTP packets. However, as discussed in [BCK] and RFC 6151 [RFC6151], this is not a secure MAC and therefore MUST be deprecated.

3. Replacement Recommendation

If NTP authentication is implemented, then AES-CMAC as specified in RFC 4493 [RFC4493] MUST be computed over all fields in the NTP header, and any extension fields that are present in the NTP packet as described in RFC 5905 [RFC5905]. The MAC key for NTP MUST be 128 bits long AES-128 key and the resulting MAC tag MUST be at least 128 bits long as stated in section 2.4 of RFC 4493 [RFC4493]. NTP makes this transition possible as it supports algorithm agility as described in Section 2.1 of RFC 7696 [RFC7696].

The hosts who wish to use NTP authentication share a symmetric key out-of-band. So they MUST implement AES-CMAC and share the corresponding symmetric key. A symmetric key is a triplet of ID, type (e.g. MD5, AES-CMAC) and the key itself. All three have to match in order to successfully authenticate packets between two hosts. Old implementations that don't support AES-CMAC will not accept and will not send packets authenticated with such a key.

4. Motivation

AES-CMAC is recommended for the following reasons:

1. It is an IETF standard that is available in many open source implementations.
2. It is immune to nonce-reuse vulnerabilities (e.g. [Joux]) because it does not use a nonce.
3. It has fine performance in terms of latency and throughput.
4. It benefits from native hardware support, for instance, Intel's New Instruction set GUE [GUE].

5. Test Vectors

For test vectors and their outputs refer to Section 4 of RFC 4493 [RFC4493]

6. Security Considerations

Refer to the Appendices A, B and C of NIST document on recommendation for the CMAC mode of authentication [NIST] and Security Considerations Section of RFC 4493 [RFC4493] for discussion on security guarantees of AES-CMAC.

7. Acknowledgements

The authors wish to acknowledge useful discussions with Leen Alshenibr, Daniel Franke, Ethan Heilman, Kenny Paterson, Leonid Reyzin, Harlan Stenn, and Mayank Varia.

8. IANA Considerations

This memo includes no request to IANA.

9. References

9.1. Normative References

- [NIST] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", <<https://www.nist.gov/publications/recommendation-block-cipher-modes-operation-cmac-mode-authentication-0>>.
- [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>>.
- [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

- [BCK] Bellare, M., Canetti, R., and H. Krawczyk, "Keyed Hash Functions and Message Authentication", in Proceedings of Crypto'96, 1996.
- [GUE] Geuron, S., "Intel Advanced Encryption Standard (AES) New Instructions Set", <<https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf>>.
- [Joux] Joux, A., "Authentication Failures in NIST version of GCM", <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf>.

- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<https://www.rfc-editor.org/info/rfc1321>>.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.
- [RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <<https://www.rfc-editor.org/info/rfc7696>>.
- [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>>.

Authors' Addresses

Aanchal Malhotra
Boston University
111 Cummington St
Boston, MA 02215
US

Email: aanchal4@bu.edu

Sharon Goldberg
Boston University
111 Cummington St
Boston, MA 02215
US

Email: goldbe@cs.bu.edu

Network Working Group
Internet-Draft
Intended status: Historic
Expires: 19 August 2022

B. Haberman, Ed.
JHU
February 2022

Control Messages Protocol for Use with Network Time Protocol Version 4
draft-ietf-ntp-mode-6-cmds-11

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 5 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Control Message Overview	3
1.2. Remote Facility Message Overview	5
2. NTP Control Message Format	5
3. Status Words	7
3.1. System Status Word	8
3.2. Peer Status Word	10
3.3. Clock Status Word	12
3.4. Error Status Word	12
4. Commands	13
5. IANA Considerations	16
6. Security Considerations	16
7. Contributors	18
8. Acknowledgements	18
9. References	18
9.1. Normative References	18
9.2. Informative References	19
Appendix A. NTP Remote Facility Message Format	20
Author's Address	22

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 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.

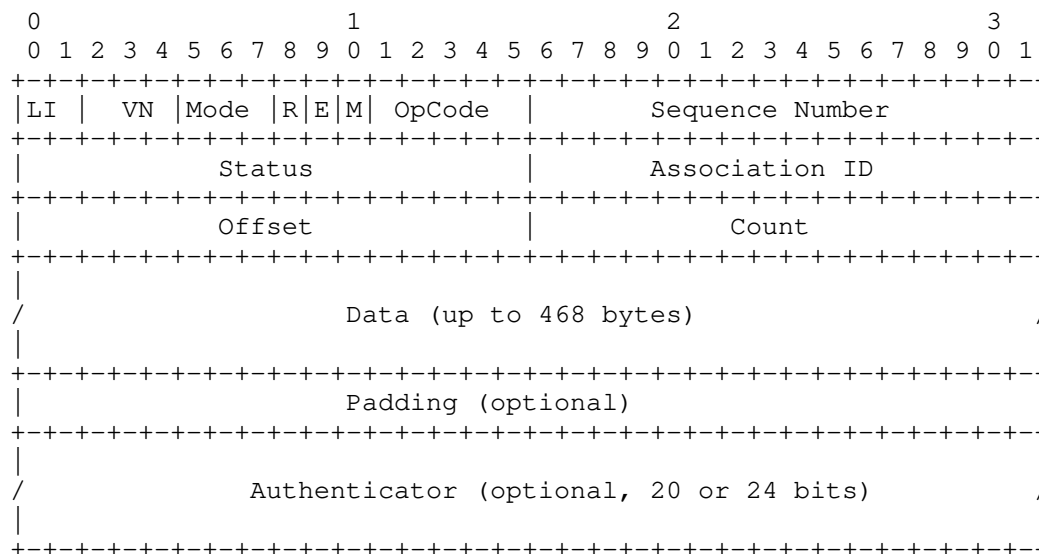


Figure 1: NTP Control Message Header

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.

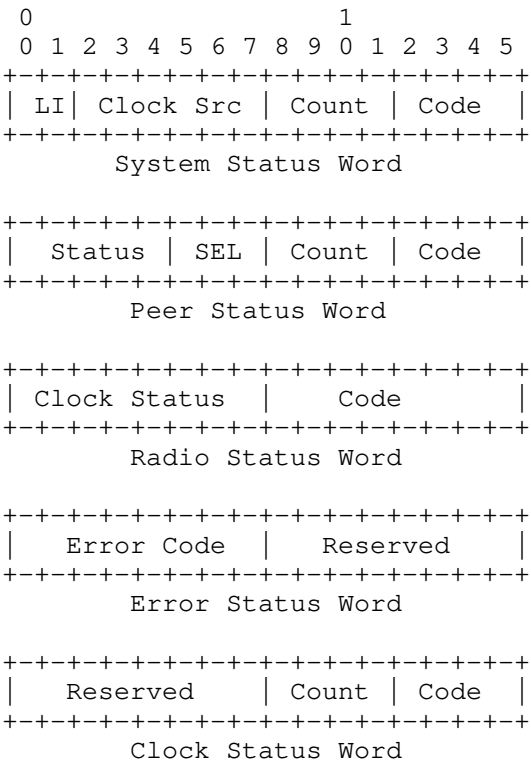


Figure 2: Status Word Formats

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. This command supports all of the state variables defined in Section 9 of [RFC5905]. 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. The use of the legacy mode 6 interface is NOT RECOMMENDED. 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). Additionally, the legacy interface for mode 6 commands SHOULD NOT be utilized in new deployments or implementation of NTP.

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>", 2 January 2014.
- [CVE-Replay] NIST National Vulnerability Database, "CVE-2015-8140, <https://nvd.nist.gov/vuln/detail/CVE-2015-8140>", 30 January 2015.
- [CVE-SHIFT] NIST National Vulnerability Database, "CVE-2016-1548, <https://nvd.nist.gov/vuln/detail/CVE-2016-1548>", 6 January 2017.
- [CVE-SPOOF] NIST National Vulnerability Database, "CVE-2015-8139, <https://nvd.nist.gov/vuln/detail/CVE-2015-8139>", 30 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.

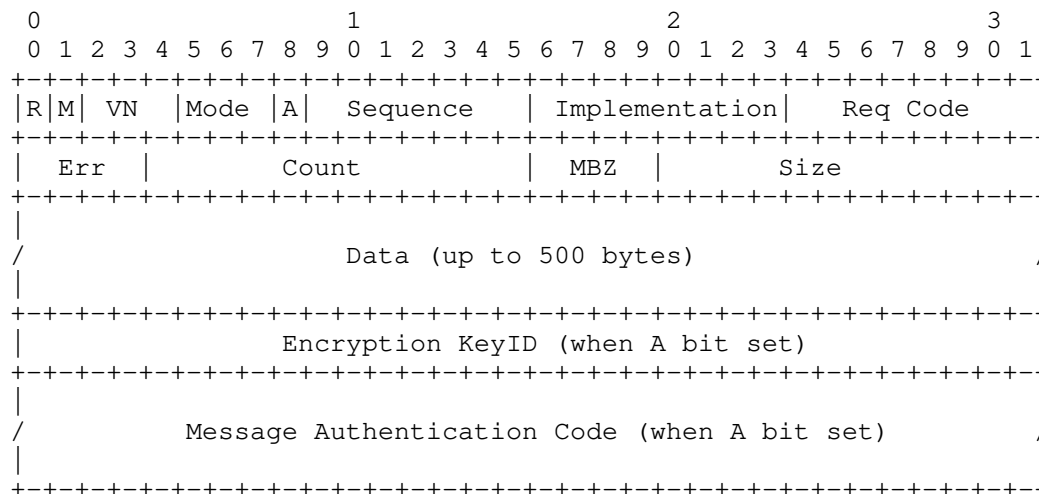


Figure 3: NTP Remote Facility Message Header

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

NTP Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2020

T. Mizrahi
Huawei Smart Platforms iLab
J. Fabini
TU Wien
A. Morton
AT&T Labs
March 11, 2020

Guidelines for Defining Packet Timestamps
draft-ietf-ntp-packet-timestamps-09

Abstract

Various network protocols make use of binary-encoded timestamps that are incorporated in the protocol packet format, referred to as packet timestamps for short. This document specifies guidelines for defining packet timestamp formats in networking protocols at various layers. It also presents three recommended timestamp formats. The target audience of this document includes network protocol designers. It is expected that a new network protocol that requires a packet timestamp will, in most cases, use one of the recommended timestamp formats. If none of the recommended formats fits the protocol requirements, the new protocol specification should specify the format of the packet timestamp according to the guidelines in this document.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 12, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Background	3
1.2. Scope of this Document	3
1.3. How to Use This Document	3
2. Terminology	4
2.1. Requirements Language	4
2.2. Abbreviations	4
2.3. Terms used in this Document	4
3. Packet Timestamp Specification Template	5
4. Recommended Timestamp Formats	6
4.1. Using a Recommended Timestamp Format	7
4.2. NTP Timestamp Formats	7
4.2.1. NTP 64-bit Timestamp Format	7
4.2.2. NTP 32-bit Timestamp Format	9
4.3. The PTP Truncated Timestamp Format	10
5. Synchronization Aspects	11
6. Timestamp Use Cases	12
6.1. Example 1	13
6.2. Example 2	14
7. Packet Timestamp Control Field	14
7.1. High-level Control Field Requirements	15
8. IANA Considerations	16
9. Security Considerations	16
10. Acknowledgments	17
11. References	17
11.1. Normative References	17
11.2. Informative References	17
Authors' Addresses	19

1. Introduction

1.1. Background

Timestamps are widely used in network protocols for various purposes: timestamps are used for logging or reporting the time of an event, delay measurement and clock synchronization protocols both make use of timestamped messages, and in security protocols a timestamp is often used as part of a value that is unlikely to repeat (nonce).

Timestamps are represented in the RFC series in one of two forms: text-based timestamps, and packet timestamps. Text-based timestamps [RFC3339] are represented as user-friendly strings, and are widely used in the RFC series, for example in information objects and data models, e.g., [RFC5646], [RFC6991], and [RFC7493]. Packet timestamps, on the other hand, are represented by a compact binary field that has a fixed size, and are not intended to have a human-friendly format. Packet timestamps are also very common in the RFC series, and are used for example for measuring delay and for synchronizing clocks, e.g., [RFC5905], [RFC4656], and [RFC7323].

1.2. Scope of this Document

This document presents guidelines for defining a packet timestamp format in network protocols. Three recommended timestamp formats are presented. It is expected that a new network protocol that requires a packet timestamp will, in most cases, use one of these recommended timestamp formats. In some cases a network protocol may use more than one of the recommended timestamp formats. However, if none of the recommended formats fits the protocol requirements, the new protocol specification should specify the format of the packet timestamp according to the guidelines in this document.

The rationale behind defining a relatively small set of recommended formats is that it enables significant reuse; network protocols can typically reuse the timestamp format of the Network Time Protocol (NTP) or the Precision Time Protocol (PTP), allowing a straightforward integration with an NTP or a PTP-based timer. Moreover, since accurate timestamping mechanisms are often implemented in hardware, a new network protocol that reuses an existing timestamp format can be quickly deployed using existing hardware timestamping capabilities.

1.3. How to Use This Document

This document is intended as a reference for network protocol designers. When defining a network protocol that uses a packet timestamp, the recommended timestamp formats should be considered

first (Section 4). If one of these formats is used, it should be referenced along the lines of the examples in Section 6.1 and Section 6.2. If none of the recommended formats fits the required functionality, then a new timestamp format should be defined using the template of Section 3.

2. Terminology

2.1. 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.

2.2. Abbreviations

NTP	Network Time Protocol [RFC5905]
PTP	Precision Time Protocol [IEEE1588]
TAI	International Atomic Time
UTC	Coordinated Universal Time

2.3. Terms used in this Document

Timestamp:	A value that represents a point in time, corresponding to an event that occurred or is scheduled to occur.
Timestamp error:	The difference between the timestamp value and the value of a reference clock at the time of the event that the timestamp was intended to indicate.
Timestamp format:	The specification of a timestamp, which is represented by a set of attributes that unambiguously define the syntax and semantics of a timestamp.
Timestamp accuracy:	The mean over an ensemble of measurements of the timestamp error.
Timestamp precision:	The variation over an ensemble of measurements of the timestamp error.

Timestamp resolution: The minimal time unit used for representing the timestamp.

3. Packet Timestamp Specification Template

This document recommends to use the timestamp formats defined in Section 4. In cases where these timestamp formats do not satisfy the protocol requirements, the timestamp specification should clearly state the reasons for defining a new format. Moreover, it is recommended to derive the new timestamp format from an existing timestamp format, either a timestamp format from this document, or any other previously defined timestamp format.

The timestamp specification must unambiguously define the syntax and the semantics of the timestamp. The current section defines the minimum set of attributes, but it should be noted that in some cases additional attributes or aspects will need to be defined in the timestamp specification.

This section defines a template for specifying packet timestamps. A timestamp format specification **MUST** include at least the following aspects:

Timestamp syntax:

- Size: The number of bits (or octets) used to represent the packet timestamp field. If the timestamp is comprised of more than one field, the size of each field is specified. Network order (big endian) is assumed by default; if this is not the case then this section explicitly specifies the endianness.

Timestamp semantics:

- Units: The units used to represent the timestamp. If the timestamp is comprised of more than one field, the units of each field are specified. If a field is limited to a specific range of values, this section specifies the permitted range of values.
- Resolution: The timestamp resolution; the resolution is equal to the timestamp field unit. If the timestamp consists of two or more fields using different time units, then the resolution is the smallest time unit.
- Wraparound: The wraparound period of the timestamp; any further wraparound-related considerations should be described here.
- Epoch: The origin of the timescale used for the timestamp; the moment in time used as a reference for the timestamp value. For

example, the epoch may be based on a standard time scale, such as UTC. Another example is a relative timestamp, in which the epoch could be the time at which the device using the timestamp was powered up, and is not affected by leap seconds (see the next attribute).

- Leap seconds: This subsection specifies whether the timestamp is affected by leap seconds. If the timestamp is affected by leap seconds, then it represents the time elapsed since the epoch minus the number of leap seconds that have occurred since the epoch.

Synchronization aspects:

The specification of a network protocol that makes use of a packet timestamp is expected to include the synchronization aspects of using the timestamp. While the synchronization aspects are not strictly part of the timestamp format specification, these aspects provide the necessary context for using the timestamp within the scope of the protocol. In some cases timestamps are used without synchronization, e.g., a timestamp that indicates the number of seconds since power up. In such cases the Synchronization Aspects section will specify that the timestamp does not correspond to a synchronized time reference, and may discuss how this affects the usage of the timestamp. Further details about synchronization aspects are discussed in Section 5.

4. Recommended Timestamp Formats

This document defines a set of recommended timestamp formats. Clearly, different network protocols may have different requirements and constraints, and consequently may use different timestamp formats. The choice of the specific timestamp format for a given protocol may depend on a various factors. A few examples of factors that may affect the choice of the timestamp format:

- o Timestamp size: while some network protocols use a large timestamp field, in some cases there may be constraints with respect to the timestamp size, affecting the choice of the timestamp format.
- o Resolution: the time resolution is another factor that may directly affect the selected timestamp format. A potentially important factor in this context is extensibility; it may be desirable to allow a timestamp format to be extensible to a higher resolution by extending the field. For example, the resolution of the NTP 32-bit timestamp format can be improved by extending it to the NTP 64-bit timestamp format in a straightforward way.

- o Wraparound period: the length of the time interval in which the timestamp is unique may also be an important factor in choosing the timestamp format. Along with the timestamp resolution, these two factors determine the required number of bits in the timestamp.
- o Common format for multiple protocols: if there are two or more network protocols that use timestamps and are often used together in typical systems, using a common timestamp format should be preferred if possible. For example, if the network protocol that is being defined typically runs on a PC, then an NTP-based timestamp format may allow easier integration with an NTP-synchronized timer. In contrast, a protocol that is typically deployed on a hardware-based platform, may make better use of a PTP-based timestamp, allowing more efficient integration with a PTP-synchronized timer.

4.1. Using a Recommended Timestamp Format

A specification that uses one of the recommended timestamp formats should specify explicitly that this is a recommended timestamp format, and point to the relevant section in the current document.

4.2. NTP Timestamp Formats

4.2.1. NTP 64-bit Timestamp Format

The Network Time Protocol (NTP) 64-bit timestamp format is defined in [RFC5905]. This timestamp format is used in several network protocols, including [RFC6374], [RFC4656], and [RFC5357]. Since this timestamp format is used in NTP, this timestamp format should be preferred in network protocols that are typically deployed in concert with NTP.

The format is presented in this section according to the template defined in Section 3.

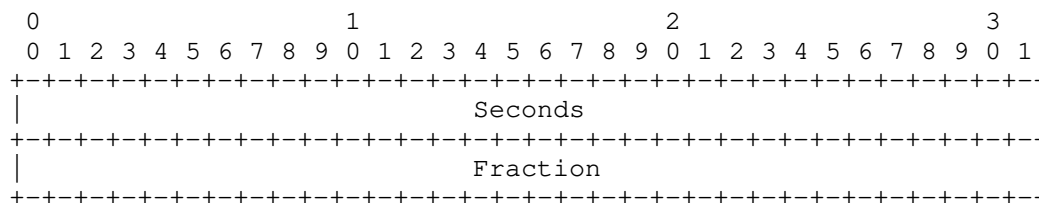


Figure 1: NTP [RFC5905] 64-bit Timestamp Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

- Size: 32 bits.

- Units: seconds.

Fraction: specifies the fractional portion of the number of seconds since the epoch.

- Size: 32 bits.

- Units: the unit is $2^{(-32)}$ seconds, which is roughly equal to 233 picoseconds.

Epoch:

The epoch is 1 January 1900 at 00:00 UTC.

Note: As pointed out in [RFC5905], strictly speaking, UTC did not exist prior to 1 January 1972, but it is convenient to assume it has existed for all eternity. The current epoch implies that the timestamp specifies the number of seconds since 1 January 1972 at 00:00 UTC plus 2272060800 (which is the number of seconds between 1 January 1900 and 1 January 1972).

Leap seconds:

This timestamp format is affected by leap seconds. The timestamp represents the number of seconds elapsed since the epoch minus the number of leap seconds. Thus, during and possibly before and/or after the occurrence of a leap second, the value of the timestamp may temporarily be ambiguous, as further discussed in Section 5.

Resolution:

The resolution is $2^{(-32)}$ seconds.

Wraparound:

This time format wraps around every 2^{32} seconds, which is roughly 136 years. The next wraparound will occur in the year 2036.

4.2.2. NTP 32-bit Timestamp Format

The Network Time Protocol (NTP) 32-bit timestamp format is defined in [RFC5905]. This timestamp format is used in [I-D.ietf-ippm-initial-registry] and [I-D.ietf-sfc-nsh-dc-allocation]. This timestamp format should be preferred in network protocols that are typically deployed in concert with NTP. The 32-bit format can be used either when space constraints do not allow the use of the 64-bit format, or when the 32-bit format satisfies the resolution and wraparound requirements.

The format is presented in this section according to the template defined in Section 3.

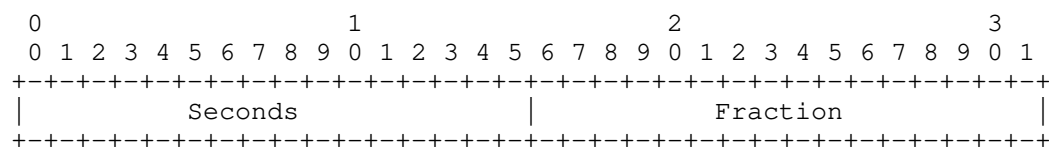


Figure 2: NTP [RFC5905] 32-bit Timestamp Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

- Size: 16 bits.

- Units: seconds.

Fraction: specifies the fractional portion of the number of seconds since the epoch.

- Size: 16 bits.

- Units: the unit is 2^{-16} seconds, which is roughly equal to 15.3 microseconds.

Epoch:

The epoch is 1 January 1900 at 00:00 UTC.

Note: As pointed out in [RFC5905], strictly speaking, UTC did not exist prior to 1 January 1972, but it is convenient to assume it has existed for all eternity. The current epoch implies that the timestamp specifies the number of seconds since 1 January 1972 at

00:00 UTC plus 2272060800 (which is the number of seconds between 1 January 1900 and 1 January 1972).

Leap seconds:

This timestamp format is affected by leap seconds. The timestamp represents the number of seconds elapsed since the epoch minus the number of leap seconds. Thus, during and possibly after the occurrence of a leap second, the value of the timestamp may temporarily be ambiguous, as further discussed in Section 5.

Resolution:

The resolution is 2^{-16} seconds.

Wraparound:

This time format wraps around every 2^{16} seconds, which is roughly 18 hours.

4.3. The PTP Truncated Timestamp Format

The Precision Time Protocol (PTP) [IEEE1588] uses an 80-bit timestamp format. The truncated timestamp format is a 64-bit field, which is the 64 least significant bits of the 80-bit PTP timestamp. Since this timestamp format is similar to the one used in PTP, this timestamp format should be preferred in network protocols that are typically deployed in PTP-capable devices.

The PTP truncated timestamp format was defined in [IEEE1588v1] and is used in several protocols, such as [RFC6374], [RFC7456], [RFC8186] and [ITU-T-Y.1731].

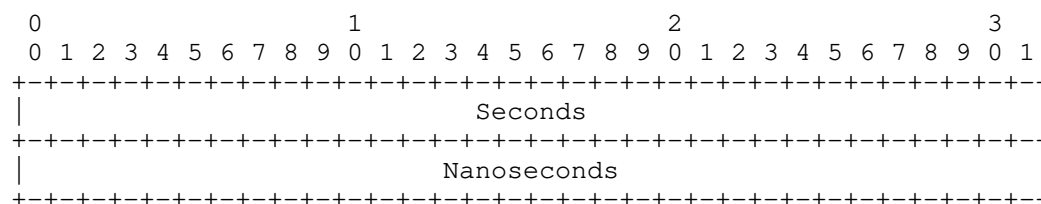


Figure 3: PTP [IEEE1588] Truncated Timestamp Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

- Size: 32 bits.

- Units: seconds.

Nanoseconds: specifies the fractional portion of the number of seconds since the epoch.

- Size: 32 bits.

- Units: nanoseconds. The value of this field is in the range 0 to $(10^9)-1$.

Epoch:

The PTP [IEEE1588] epoch is 1 January 1970 00:00:00 TAI.

Leap seconds:

This timestamp format is not affected by leap seconds.

Resolution:

The resolution is 1 nanosecond.

Wraparound:

This time format wraps around every 2^{32} seconds, which is roughly 136 years. The next wraparound will occur in the year 2106.

5. Synchronization Aspects

A specification that defines a new timestamp format or uses one of the recommended timestamp formats should include a section on Synchronization Aspects. Note that the recommended timestamp formats defined in this document (Section 4) do not include the synchronization aspects of these timestamp formats, but it is expected that specifications of network protocols that make use of these formats should include the synchronization aspects. Examples of a Synchronization Aspects section can be found in Section 6.

The Synchronization Aspects section should specify all the assumptions and requirements related to synchronization. For example, the synchronization aspects may specify whether nodes populating the timestamps should be synchronized among themselves, and whether the timestamp is measured with respect to a central reference clock such as an NTP server. If time is assumed to be synchronized to a time standard such as UTC or TAI, it should be specified in this section. Further considerations may be discussed

in this section, such as the required timestamp accuracy and precision.

Another aspect that should be discussed in this section is leap second [RFC5905] considerations. The timestamp specification template (Section 3) specifies whether the timestamp is affected by leap seconds. It is often the case that further details about leap seconds will need to be defined in the Synchronization Aspects section. Generally speaking, a leap second is a one-second adjustment that is occasionally applied to UTC in order to keep it aligned to the solar time. A leap second may be either positive or negative, i.e., the clock may either be shifted one second forwards or backwards. All leap seconds that have occurred up to the publication of this document have been in the backwards direction, and although forward leap seconds are theoretically possible, the text throughout this document focuses on the common case, which is the backward leap second. In a timekeeping system that considers leap seconds, the system clock may be affected by a leap second in one of three possible ways:

- o The clock is turned backwards one second at the end of the leap second.
- o The clock is frozen during the duration of the leap second.
- o The clock is slowed down during the leap second and adjacent time intervals until the new time value catches up. The interval for this process, commonly referred to as leap smear, can range from several seconds to several hours before, during, and/or after the occurrence of the leap second.

The way leap seconds are handled depends on the synchronization protocol, and is thus not specified in this document. However, if a timestamp format is defined with respect to a timescale that is affected by leap seconds, the Synchronization Aspects section should specify how the use of leap seconds affects the timestamp usage.

6. Timestamp Use Cases

Packet timestamps are used in various network protocols. Typical applications of packet timestamps include delay measurement, clock synchronization, and others. The following table presents a (non-exhaustive) list of protocols that use packet timestamps, and the timestamp formats used in each of these protocols.

Protocol	Recommended formats			Other
	NTP 64-bit	NTP 32-bit	PTP Trunc.	
NTP [RFC5905]	+			
OWAMP [RFC4656]	+			
TWAMP [RFC5357]	+		+	
TWAMP [RFC8186]	+			
TRILL [RFC7456]			+	
MPLS [RFC6374]			+	
TCP [RFC7323]				+
RTP [RFC3550]	+			+
IPFIX [RFC7011]				+
BinaryTime [RFC6019]				+
[I-D.ietf-ippm-initial-registry]	+	+		
[I-D.ietf-sfc-nsh-dc-allocation]		+	+	

Figure 4: Protocols that use Packet Timestamps

The rest of this section presents two hypothetical examples of network protocol specifications that use one of the recommended timestamp formats. The examples include the text that specifies the information related to the timestamp format.

6.1. Example 1

Timestamp:

The timestamp format used in this specification is the NTP [RFC5905] 64-bit format, as specified in Section 4.2.1 of [I-D.ietf-ntp-packet-timestamps].

Synchronization aspects:

It is assumed that nodes that run this protocol are synchronized to UTC using a synchronization mechanism that is outside the scope of this document. In typical deployments this protocol will run on a machine that uses NTP [RFC5905] for synchronization. Thus, the timestamp may be derived from the NTP-synchronized clock, allowing the timestamp to be measured with respect to the clock of an NTP server. Since the NTP time format is affected by leap seconds, the current timestamp format is similarly affected. Thus, the value of a timestamp during or slightly after a leap second may be temporarily inaccurate.

6.2. Example 2

Timestamp:

The timestamp format used in this specification is the PTP [IEEE1588] Truncated format, as specified in Section 4.3 of [I-D.ietf-ntp-packet-timestamps].

Synchronization aspects:

It is assumed that nodes that run this protocol are synchronized among themselves. Nodes may be synchronized to a global reference time. Note that if PTP [IEEE1588] is used for synchronization, the timestamp may be derived from the PTP-synchronized clock, allowing the timestamp to be measured with respect to the clock of an PTP Grandmaster clock.

7. Packet Timestamp Control Field

In some cases it is desirable to have a control field that describes structure, format, content, and properties of timestamps. Control information about the timestamp format can be conveyed in some protocols using a dedicated control plane protocol, or may be made available at the management plane, for example using a YANG data model. An optional control field allows some of the control information to be attached to the timestamp.

An example of a packet timestamp control field is the Error Estimate field, defined by Section 4.1.2 in [RFC4656], which is used in OWAMP [RFC4656] and TWAMP [RFC5357]. The Root Dispersion and Root Delay fields in the NTP header [RFC5905] are two examples of fields that provide information about the timestamp precision. Another example of an auxiliary field is the Correction Field in the PTP header [IEEE1588]; its value is used as a correction to the timestamp, and may be assigned by the sender of the PTP message and updated by transit nodes (Transparent Clocks) in order to account for the delay along the path.

This section defines high-level guidelines for defining packet timestamp control fields in network protocols that can benefit from such timestamp-related control information. The word 'requirements' is used in its informal context in this section.

7.1. High-level Control Field Requirements

A control field for packet timestamps must offer an adequate feature set and fulfill a series of requirements to be usable and accepted. The following list captures the main high-level requirements for timestamp fields.

1. **Extensible Feature Set:** protocols and applications depend on various timestamp characteristics. A timestamp control field must support a variable number of elements (components) that either describe or quantify timestamp-specific characteristics or parameters. Examples of potential elements include timestamp size, encoding, accuracy, leap seconds, reference clock identifiers, etc.
2. **Size:** Essential for an efficient use of timestamp control fields is the trade-off between supported features and control field size. Protocols and applications may select the specific control field elements that are needed for their operation from the set of available elements.
3. **Composition:** Applications may depend on specific control field elements being present in messages. The status of these elements may be either mandatory, conditional mandatory, or optional, depending on the specific application and context. A control field specification must support applications in conveying or negotiating (a) the set of control field elements along with (b) the status of any element (i.e., mandatory, conditional mandatory, or optional) by defining appropriate data structures and identity codes.
4. **Category:** Control field elements can characterize either static timestamp information (like, e.g., timestamp size in bytes and timestamp semantics: NTP 64 bit format) or runtime timestamp information (like, e.g., estimated timestamp accuracy at the time of sampling: 20 microseconds to UTC). For efficiency reason it may be meaningful to support separation of these two concepts: while the former (static) information is typically valid throughout a protocol session and may be conveyed only once, at session establishment time, the latter (runtime) information augments any timestamp instance and may cause substantial overhead for high-traffic protocols.

Proposals for timestamp control fields will be defined in separate documents and are out of scope of this document.

8. IANA Considerations

This document includes no request to IANA.

9. Security Considerations

A network protocol that uses a packet timestamp MUST specify the security considerations that result from using the timestamp. This section provides an overview of some of the common security considerations of using timestamps.

Any metadata that is attached to control or data packets, and specifically packet timestamps, can facilitate network reconnaissance; by passively eavesdropping to timestamped packets an attacker can gather information about the network performance, and about the level of synchronization between nodes.

In some cases timestamps could be spoofed or modified by on-path attackers, thus attacking the application that uses the timestamps. For example, if timestamps are used in a delay measurement protocol, an attacker can modify en route timestamps in a way that manipulates the measurement results. Integrity protection mechanisms, such as Message Authentication Codes (MAC), can mitigate such attacks. The specification of an integrity protection mechanism is outside the scope of this document, as typically integrity protection will be defined on a per-network-protocol basis, and not specifically for the timestamp field.

Another potential threat that can have a similar impact is delay attacks. An attacker can maliciously delay some or all of the en route messages, with the same harmful implications as described in the previous paragraph. Mitigating delay attacks is a significant challenge; in contrast to spoofing and modification attacks, the delay attack cannot be prevented by cryptographic integrity protection mechanisms. In some cases delay attacks can be mitigated by sending the timestamped information through multiple paths, allowing to detect and to be resilient to an attacker that has access to one of the paths.

In many cases timestamping relies on an underlying synchronization mechanism. Thus, any attack that compromises the synchronization mechanism can also compromise protocols that use timestamping. Attacks on time protocols are discussed in detail in [RFC7384].

10. Acknowledgments

The authors thank Russ Housley, Yaakov Stein, Greg Mirsky, Warner Losh, Rodney Cummings, Miroslav Lichvar, Denis Reilly, Daniel Franke, Eric Vyncke, Ben Kaduk, Ian Swett, Francesca Palombini, Watson Ladd, and other members of the NTP working group for many helpful comments. The authors gratefully acknowledge Harlan Stenn and the people from the Network Time Foundation for sharing their thoughts and ideas.

11. References

11.1. 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>>.

11.2. Informative References

- [I-D.ietf-ippm-initial-registry] Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metrics Registry Entries", draft-ietf-ippm-initial-registry-16 (work in progress), March 2020.
- [I-D.ietf-ntp-packet-timestamps] Mizrahi, T., Fabini, J., and A. Morton, "Guidelines for Defining Packet Timestamps", draft-ietf-ntp-packet-timestamps-08 (work in progress), February 2020.
- [I-D.ietf-sfc-nsh-dc-allocation] Guichard, J., Smith, M., Kumar, S., Majee, S., and T. Mizrahi, "Network Service Header (NSH) MD Type 1: Context Header Allocation (Data Center)", draft-ietf-sfc-nsh-dc-allocation-02 (work in progress), September 2018.
- [IEEE1588] IEEE, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", 2008.

- [IEEE1588v1] IEEE, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2002.
- [ITU-T-Y.1731] ITU-T, "OAM functions and mechanisms for Ethernet based Networks", 2013.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <<https://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [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>>.
- [RFC6019] Housley, R., "BinaryTime: An Alternate Format for Representing Date and Time in ASN.1", RFC 6019, DOI 10.17487/RFC6019, September 2010, <<https://www.rfc-editor.org/info/rfc6019>>.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<https://www.rfc-editor.org/info/rfc6374>>.

- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <<https://www.rfc-editor.org/info/rfc7323>>.
- [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>>.
- [RFC7456] Mizrahi, T., Senevirathne, T., Salam, S., Kumar, D., and D. Eastlake 3rd, "Loss and Delay Measurement in Transparent Interconnection of Lots of Links (TRILL)", RFC 7456, DOI 10.17487/RFC7456, March 2015, <<https://www.rfc-editor.org/info/rfc7456>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC8186] Mirsky, G. and I. Meilik, "Support of the IEEE 1588 Timestamp Format in a Two-Way Active Measurement Protocol (TWAMP)", RFC 8186, DOI 10.17487/RFC8186, June 2017, <<https://www.rfc-editor.org/info/rfc8186>>.

Authors' Addresses

Tal Mizrahi
Huawei Smart Platforms iLab
8-2 Matam
Haifa 3190501
Israel

Email: tal.mizrahi.phd@gmail.com

Joachim Fabini
TU Wien
Gusshausstrasse 25/E389
Vienna 1040
Austria

Phone: +43 1 58801 38813
Fax: +43 1 58801 38898
Email: Joachim.Fabini@tuwien.ac.at
URI: <http://www.tc.tuwien.ac.at/about-us/staff/joachim-fabini/>

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 26, 2019

H. Stenn
Network Time Foundation
S. Goldberg
Boston University
March 25, 2019

Network Time Protocol REFID Updates
draft-ietf-ntp-refid-updates-05

Abstract

RFC 5905 [RFC5905], section 7.3, "Packet Header Variables", defines the value of the REFID, the system peer for the responding host. In the past, for IPv4 associations the IPv4 address is used, and for IPv6 associations the first four octets of the MD5 hash of the IPv6 address are used. There are two recognized shortcomings to this approach, and this proposal addresses them. One is that knowledge of the system peer is "abusable" information and should not be generally available. The second is that the four octet hash of the IPv6 address looks very much like an IPv4 address, and this is confusing.

RFC EDITOR: PLEASE REMOVE THE FOLLOWING PARAGRAPH BEFORE PUBLISHING:

The source code and issues list for this draft can be found in <https://github.com/hstenn/ietf-ntp-refid-updates>

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 26, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. The REFID	2
1.2. NOT-YOU REFID	3
1.3. IPv6 REFID	4
1.4. Requirements Language	4
2. The NOT-YOU REFID	4
2.1. Proposal	5
3. Augmenting the IPv6 REFID Hash	5
3.1. Background	5
3.2. Potential Problems	6
4. Acknowledgements	6
5. IANA Considerations	6
6. Security Considerations	6
7. References	7
7.1. Normative References	7
7.2. Informative References	7
Authors' Addresses	7

1. Introduction

1.1. The REFID

The interpretation of a REFID is based on the stratum, as documented in RFC 5905 [RFC5905], section 7.3, "Packet Header Variables". The core reason for the REFID in the NTP Protocol is to prevent a degree-one timing loop, where server B decides to follow A as its time source, and A then decides to follow B as its time source.

At Stratum 2+, which will be the case if two servers A and B are exchanging timing information, then if server B follows A as its time source, A's address will be B's REFID. When A uses IPv4, the default

REFID is A's IPv4 address. When A uses IPv6, the default REFID is a four-octet digest of A's IPv6 address. Now, if A queries B for its time, then A will learn that B is using A as its time source by observing A's address in the REFID field of the response packet sent by B. Thus, A will not select B as a potential time source, as this would cause a timing loop.

1.2. NOT-YOU REFID

The traditional REFID mechanism, however, also allows a third-party C to learn that A is the time source that is being used by B. When A is using IPv4, C can learn this by querying B for its time, and observing that the REFID in B's response is the IPv4 address of A. Meanwhile, when A is using IPv6, then C can again query B for its time, and then can use an offline dictionary attack to attempt to determine the IPv6 address that corresponds to the digest value in the response sent by B. C could construct the necessary dictionary by compiling a list of publicly accessible IPv6 servers. Remote attackers can use this technique to attempt to identify the time sources used by a target, and then send spoofed packets to the target or its time source in an attempt to disrupt time service, as was done e.g., in [NDSS16] or [CVE-2015-8138].

The REFID thus unnecessarily leaks information about a target's time server to remote attackers. The best way to mitigate this vulnerability is to decouple the IP address of the time source from the REFID. To do this, a system can use an otherwise-impossible value for its REFID, called the NOT-YOU REFID value, when it believes that a querying system is not its time source.

The NOT-YOU REFID proposal is backwards-compatible and provides the bare minimum diagnostic information to third parties. It can be implemented by one peer in an NTP association without any changes to the other peer. This holds as long as responding NOT-YOU system can accurately detect when it's getting a request from its system peer.

The NOT-YOU REFID proposal does have a small risk. Consider system A that returns the NOT-YOU REFID and system B that has two network interfaces B1 and B2. Suppose that system A is using system B as his time source, via network interface B1. Now suppose that system B queries system A for time via network interface B2. In this case, system A returns the NOT-YOU REFID value to system B, since system A does not realize that network interface B1 and B2 belong to the same system. In this case, system B might choose system A as its time source, and a degree-one timing loop will occur. In this case, however, the two systems will spiral into degrading stratum positions with increasing root distances, and eventually the loop will break. If any other systems are available as time servers, one of them will

become the new system peer. However, unless or until this happens the two spiraling systems will have degraded time quality.

1.3. IPv6 REFID

In an environment where all time queries made to a server can be trusted, an operator might well choose to expose the real REFID. RFC 5905 [RFC5905], section 7.3, "Packet Header Variables", explains how a remote system peer is converted to a REFID. It says:

If using the IPv4 address family, the identifier is the four-octet IPv4 address. If using the IPv6 family, it is the first four octets of the MD5 hash of the IPv6 address. ...

However, the MD5 hash of an IPv6 address often looks like a valid IPv4 address. When this happens, an operator cannot tell if the REFID refers to an IPv6 address or and IPv4. Specifically, the NTP Project has received a report where the generated IPv6 hash decoded to the IPv4 address of a different machine on the system peer's network.

This proposal offers a way for a system to generate a REFID for a IPv6 system peer that does not conflict with an IPv4-based REFID.

This proposal is not backwards-compatible. It SHOULD be implemented by both peers in an NTP association. In the scenario where A and B are peering using IPv6, where A is the system peer and does not understand IPv6 REFID, and B is subordinate and is using IPv6 REFID, A will not be able to determine that B is using A as its system peer and a degree-one timing loop can form.

If both peers implement the IPv6 REFID this situation cannot happen.

If at least one of the peers implements the proposed I-DO [DRAFT-I-DO] protocol this situation cannot happen.

1.4. Requirements Language

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

2. The NOT-YOU REFID

2.1. Proposal

When enabled, this proposal allows the one-degree loop detection to work and useful diagnostic information to be provided to trusted partners while keeping potentially abusable information from being disclosed to ostensibly uninterested parties. It does this by returning the normal REFID to queries that come from trusted addresses or from an address that the current system believes is its time source (aka its "system peer"), and otherwise returning one of two special IP addresses that is interpreted to mean "not you". The "not you" IP addresses are 127.127.127.127 and 127.127.127.128. If an IPv6 query is received from an address whose four-octet hash equals one of these two addresses and we believe the querying host is not our system peer, the other NOT-YOU address is returned as the REFID.

This mechanism is correct and transparent when the system responding with a NOT-YOU can accurately detect when it's getting a timing query from its system peer. A querying system that uses IPv4 continues to check that its IPv4 address does not appear in the REFID before deciding whether to take time from the current system. A querying system that uses IPv6 continues to check that the four-octet hash of its IPv6 address does not appear in the REFID before deciding whether to take time from the current system.

3. Augmenting the IPv6 REFID Hash

3.1. Background

In a trusted network, the S2+ REFID is generated based on the network system peer. RFC 5905 [RFC5905] says:

If using the IPv4 address family, the identifier is the four-octet IPv4 address. If using the IPv6 family, it is the first four octets of the MD5 hash of the IPv6 address.

This means that the IPv4 representation of the IPv6 hash would be: b1.b2.b3.b4 . This proposal is that the system MAY also use 255.b2.b3.b4 as its REFID. This reduces the risk of ambiguity, since addresses beginning with 255 are "reserved", and thus will not collide with valid IPv4 on the network.

When using the REFID to check for a timing loop for an IPv6 association, if the code that checks the first four-octets of the hash fails to match then the code must check again, using 0xFF as the first octet of the hash.

3.2. Potential Problems

There is a 1 in 16,777,216 chance that the REFID hashes of two IPv6 addresses will be identical, producing a false-positive loop detection. With a sufficient number of servers, the risk of this problem becomes a non-issue. The use of the NOT-YOU REFID and/or the proposed REFID-SUGGESTION [DRAFT-REFID-SUGGESTION] or I-DO [DRAFT-I-DO] extension fields are ways to mitigate this potential situation.

Unrealistically, if only two instances of NTP are communicating via IPv6 and system A implements this new IPv6 REFID hash and system B does not, system B will not be able to detect this loop condition. In this case, the two machines will slowly increase their stratum until they become unsynchronized. This situation is considered to be unrealistic because, for this to happen, each system would have to have only the other system available as a time source, for example, in a misconfigured "orphan mode" setup. There is no risk of this happening in an NTP network with 3 or more time sources, or in a properly-configured "time island" setup.

4. Acknowledgements

For the "not-you" REFID, we acknowledge useful discussions with Aanchal Malhotra and Matthew Van Gundy.

For the IPv6 REFID, we acknowledge Dan Mahoney (and perhaps others) for suggesting the idea of using an "impossible" first-octet value to indicate an IPv6 refid hash.

5. IANA Considerations

This memo requests IANA to allocate a pseudo Extension Field Type of 0xFFFF so the proposed "I-Do" exchange can report whether or not the "IPv6 REFID Hash" is supported.

6. Security Considerations

Many systems running NTP are configured to return responses to timing queries by default. These responses contain a REFID field, which generally reveals the address of the system's time source if that source is an IPv4 address. This behavior can be exploited by remote attackers who wish to first learn the address of a target's time source, and then attack the target and/or its time source. As such, the NOT-YOU REFID proposal is designed to harden NTP against these attacks by limiting the amount of information leaked in the REFID field.

Systems running NTP should reveal the identity of their system in peer in their REFID only when they are on a trusted network. The IPv6 REFID proposal provides one way to do this, when the system peer uses addresses in the IPv6 family.

7. References

7.1. 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>>.

7.2. Informative References

- [CVE-2015-8138] Van Gundy, M. and J. Gardner, "Network Time Protocol Origin Timestamp Check Impersonation Vulnerability (CVE-2015-8138)", in TALOS VULNERABILITY REPORT (TALOS-2016-0077), 2016.
- [DRAFT-I-DO] Stenn, H., "draft-stenn-ntp-i-do", 2018.
- [DRAFT-REFID-SUGGESTION] Stenn, H., "draft-stenn-ntp-suggest-refid", 2018.
- [NDSS16] Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", in ISOC Network and Distributed System Security Symposium 2016 (NDSS'16), 2016.
- [NTP-EXTENSION-FIELD] Stenn, H., "draft-stenn-ntp-extension-fields", 2018.

Authors' Addresses

Harlan Stenn
Network Time Foundation
P.O. Box 918
Talent, OR 97540
US

Email: stenn@nwttime.org

Sharon Goldberg
Boston University
111 Cummington St
Boston, MA 02215
US

Email: goldbe@cs.bu.edu

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 26, 2020

D. Franke
Akamai
D. Sibold
K. Teichel
PTB
M. Dansarie

R. Sundblad
Netnod
March 25, 2020

Network Time Security for the Network Time Protocol
draft-ietf-ntp-using-nts-for-ntp-28

Abstract

This memo specifies Network Time Security (NTS), a mechanism for using Transport Layer Security (TLS) and Authenticated Encryption with Associated Data (AEAD) to provide cryptographic security for the client-server mode of the Network Time Protocol (NTP).

NTS is structured as a suite of two loosely coupled sub-protocols. The first (NTS-KE) handles initial authentication and key establishment over TLS. The second handles encryption and authentication during NTP time synchronization via extension fields in the NTP packets, and holds all required state only on the client via opaque cookies.

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 26, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
1.1. Objectives	4
1.2. Protocol Overview	5
2. Requirements Language	7
3. TLS profile for Network Time Security	7
4. The NTS Key Establishment Protocol	8
4.1. NTS-KE Record Types	10
4.1.1. End of Message	11
4.1.2. NTS Next Protocol Negotiation	11
4.1.3. Error	11
4.1.4. Warning	12
4.1.5. AEAD Algorithm Negotiation	12
4.1.6. New Cookie for NTPv4	13
4.1.7. NTPv4 Server Negotiation	13
4.1.8. NTPv4 Port Negotiation	14
4.2. Retry Intervals	14
4.3. Key Extraction (generally)	15
5. NTS Extension Fields for NTPv4	15
5.1. Key Extraction (for NTPv4)	15
5.2. Packet Structure Overview	16
5.3. The Unique Identifier Extension Field	16
5.4. The NTS Cookie Extension Field	17
5.5. The NTS Cookie Placeholder Extension Field	17
5.6. The NTS Authenticator and Encrypted Extension Fields Extension Field	17
5.7. Protocol Details	20
6. Suggested Format for NTS Cookies	24
7. IANA Considerations	25
7.1. Service Name and Transport Protocol Port Number Registry	25
7.2. TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs Registry	26

7.3.	TLS Exporter Labels Registry	26
7.4.	NTP Kiss-o'-Death Codes Registry	26
7.5.	NTP Extension Field Types Registry	26
7.6.	Network Time Security Key Establishment Record Types Registry	27
7.7.	Network Time Security Next Protocols Registry	28
7.8.	Network Time Security Error and Warning Codes Registries	29
8.	Implementation Status - RFC EDITOR: REMOVE BEFORE PUBLICATION	30
8.1.	Implementation 1	30
8.1.1.	Coverage	30
8.1.2.	Licensing	31
8.1.3.	Contact Information	31
8.1.4.	Last Update	31
8.2.	Implementation 2	31
8.2.1.	Coverage	31
8.2.2.	Licensing	31
8.2.3.	Contact Information	31
8.2.4.	Last Update	31
8.3.	Implementation 3	32
8.3.1.	Coverage	32
8.3.2.	Licensing	32
8.3.3.	Contact Information	32
8.3.4.	Last Update	32
8.4.	Implementation 4	32
8.4.1.	Coverage	32
8.4.2.	Licensing	33
8.4.3.	Contact Information	33
8.4.4.	Last Update	33
8.5.	Implementation 5	33
8.5.1.	Coverage	33
8.5.2.	Licensing	33
8.5.3.	Contact Information	33
8.5.4.	Last Update	33
8.6.	Implementation 6	33
8.6.1.	Coverage	34
8.6.2.	Licensing	34
8.6.3.	Contact Information	34
8.6.4.	Last Update	34
8.7.	Interoperability	34
9.	Security Considerations	34
9.1.	Protected Modes	34
9.2.	Cookie Encryption Key Compromise	35
9.3.	Sensitivity to DDoS Attacks	35
9.4.	Avoiding DDoS Amplification	35
9.5.	Initial Verification of Server Certificates	36
9.6.	Delay Attacks	37
9.7.	NTS Stripping	38
10.	Privacy Considerations	38

10.1. Unlinkability	38
10.2. Confidentiality	39
11. Acknowledgements	39
12. References	39
12.1. Normative References	39
12.2. Informative References	41
Appendix A. Terms and Abbreviations	42
Authors' Addresses	43

1. Introduction

This memo specifies Network Time Security (NTS), a cryptographic security mechanism for network time synchronization. A complete specification is provided for application of NTS to the client-server mode of the Network Time Protocol (NTP) [RFC5905].

1.1. Objectives

The objectives of NTS are as follows:

- o Identity: Through the use of a X.509 public key infrastructure, implementations can cryptographically establish the identity of the parties they are communicating with.
- o Authentication: Implementations can cryptographically verify that any time synchronization packets are authentic, i.e., that they were produced by an identified party and have not been modified in transit.
- o Confidentiality: Although basic time synchronization data is considered non-confidential and sent in the clear, NTS includes support for encrypting NTP extension fields.
- o Replay prevention: Client implementations can detect when a received time synchronization packet is a replay of a previous packet.
- o Request-response consistency: Client implementations can verify that a time synchronization packet received from a server was sent in response to a particular request from the client.
- o Unlinkability: For mobile clients, NTS will not leak any information additional to NTP which would permit a passive adversary to determine that two packets sent over different networks came from the same client.
- o Non-amplification: Implementations (especially server implementations) can avoid acting as distributed denial-of-service

(DDoS) amplifiers by never responding to a request with a packet larger than the request packet.

- o Scalability: Server implementations can serve large numbers of clients without having to retain any client-specific state.
- o Performance: NTS must not significantly degrade the quality of the time transfer. The encryption and authentication used when actually transferring time should be lightweight (see RFC 7384, Section 5.7 [RFC7384]).

1.2. Protocol Overview

The Network Time Protocol includes many different operating modes to support various network topologies (see RFC 5905, Section 3 [RFC5905]). In addition to its best-known and most-widely-used client-server mode, it also includes modes for synchronization between symmetric peers, a control mode for server monitoring and administration, and a broadcast mode. These various modes have differing and partly contradictory requirements for security and performance. Symmetric and control modes demand mutual authentication and mutual replay protection. Additionally, for certain message types control mode may require confidentiality as well as authentication. Client-server mode places more stringent requirements on resource utilization than other modes, because servers may have vast number of clients and be unable to afford to maintain per-client state. However, client-server mode also has more relaxed security needs, because only the client requires replay protection: it is harmless for stateless servers to process replayed packets. The security demands of symmetric and control modes, on the other hand, are in conflict with the resource-utilization demands of client-server mode: any scheme which provides replay protection inherently involves maintaining some state to keep track of what messages have already been seen.

This memo specifies NTS exclusively for the client-server mode of NTP. To this end, NTS is structured as a suite of two protocols:

The "NTS Extensions for NTPv4" define a collection of NTP extension fields for cryptographically securing NTPv4 using previously-established key material. They are suitable for securing client-server mode because the server can implement them without retaining per-client state. All state is kept by the client and provided to the server in the form of an encrypted cookie supplied with each request. On the other hand, the NTS Extension Fields are suitable *only* for client-server mode because only the client, and not the server, is protected from replay.

The "NTS Key Establishment" protocol (NTS-KE) is a mechanism for establishing key material for use with the NTS Extension Fields for NTPv4. It uses TLS to establish keys, provide the client with an initial supply of cookies, and negotiate some additional protocol options. After this, the TLS channel is closed with no per-client state remaining on the server side.

The typical protocol flow is as follows: The client connects to an NTS-KE server on the NTS TCP port and the two parties perform a TLS handshake. Via the TLS channel, the parties negotiate some additional protocol parameters and the server sends the client a supply of cookies along with an address and port of an NTP server for which the cookies are valid. The parties use TLS key export [RFC5705] to extract key material which will be used in the next phase of the protocol. This negotiation takes only a single round trip, after which the server closes the connection and discards all associated state. At this point the NTS-KE phase of the protocol is complete. Ideally, the client never needs to connect to the NTS-KE server again.

Time synchronization proceeds with the indicated NTP server. The client sends the server an NTP client packet which includes several extension fields. Included among these fields are a cookie (previously provided by the key establishment server) and an authentication tag, computed using key material extracted from the NTS-KE handshake. The NTP server uses the cookie to recover this key material and send back an authenticated response. The response includes a fresh, encrypted cookie which the client then sends back in the clear in a subsequent request. (This constant refreshing of cookies is necessary in order to achieve NTS's unlinkability goal.)

Figure 1 provides an overview of the high-level interaction between the client, the NTS-KE server, and the NTP server. Note that the cookies' data format and the exchange of secrets between NTS-KE and NTP servers are not part of this specification and are implementation dependent. However, a suggested format for NTS cookies is provided in Section 6.

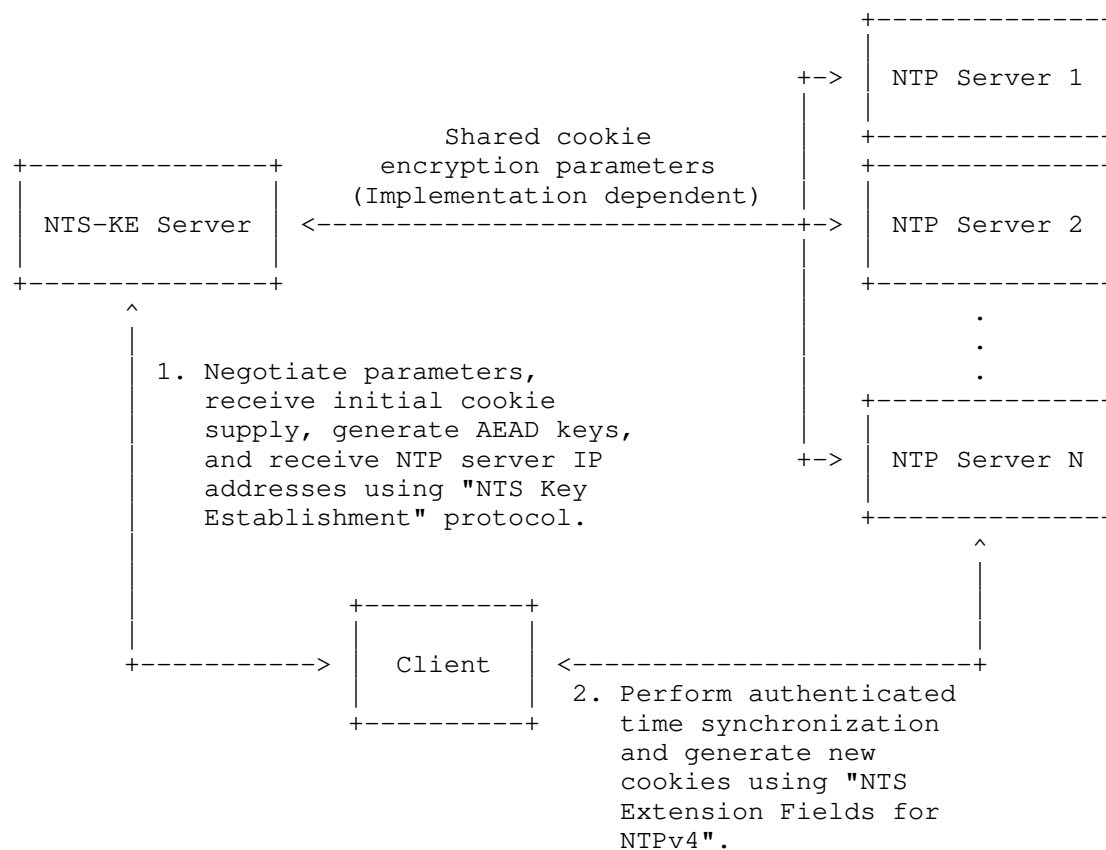


Figure 1: Overview of High-Level Interactions in NTS

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. TLS profile for Network Time Security

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 which 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.

Implementations MUST follow the rules in RFC 5280 [RFC5280] and RFC 6125 [RFC6125] for the representation and verification of the application's service identity. When NTS-KE service discovery (out of scope for this document) produces one or more host names, use of the DNS-ID identifier type [RFC6125] is RECOMMENDED; specifications for service discovery mechanisms can provide additional guidance for certificate validation based on the results of discovery. Section 9.5 of this memo discusses particular considerations for certificate verification in the context of NTS.

4. The NTS Key Establishment Protocol

The NTS key establishment protocol is conducted via TCP port [[TBD1]]. The two endpoints carry out a TLS handshake in conformance with Section 3, with the client offering (via an ALPN [RFC7301] extension), 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 RFC 8446, Section 6.1 [RFC8446].

The client's request and the server's response each SHALL consist of a sequence of records formatted according to Figure 2. 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.

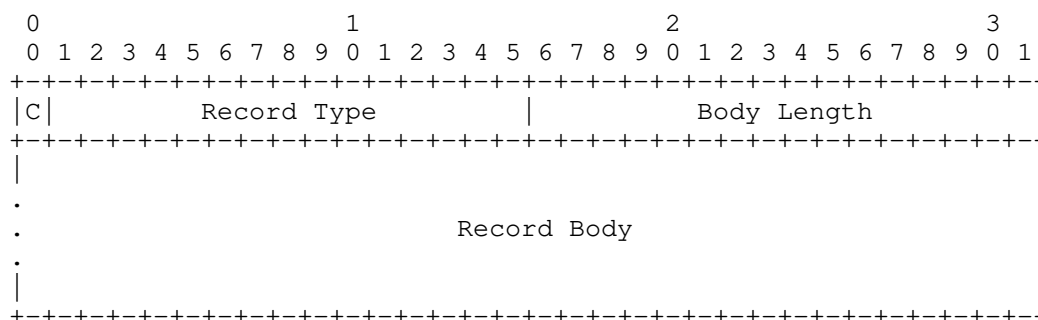


Figure 2: NTS-KE Record Format

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]'.

Note that, although the Type-Length-Body format of an NTS-KE record is similar to that of an NTP extension field, the semantics of the length field differ. While the length subfield of an NTP extension field gives the length of the entire extension field including the type and length subfields, the length field of an NTS-KE record gives just the length of the body.

Figure 3 provides a schematic overview of the key establishment. It displays the protocol steps to be performed by the NTS client and server and record types to be exchanged.

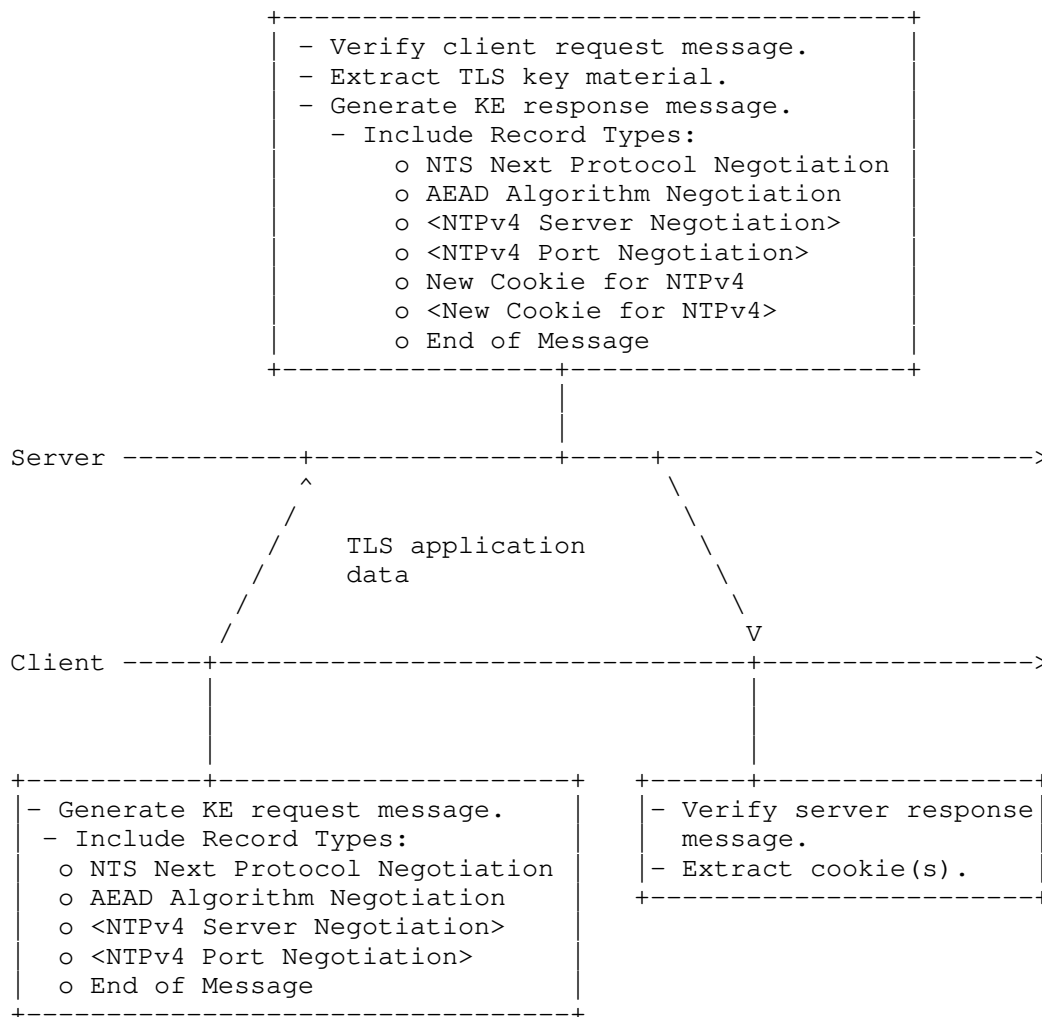


Figure 3: NTS Key Establishment Messages

4.1. NTS-KE Record Types

The following NTS-KE Record Types are defined:

4.1.1. End of Message

The End of Message record has a Record Type number of 0 and a zero-length body. It MUST occur exactly once as the final record of every NTS-KE request and response. The Critical Bit MUST be set.

4.1.2. NTS Next Protocol Negotiation

The NTS Next Protocol Negotiation record has a Record Type number of 1. It MUST occur exactly once in every NTS-KE request and response. Its body consists of a sequence of 16-bit unsigned integers in network byte order. Each integer represents a Protocol ID from the IANA Network Time Security Next Protocols registry. The Critical Bit MUST be set.

The Protocol IDs listed in the client's NTS Next Protocol Negotiation record denote those protocols which the client wishes to speak using the key material established through this NTS-KE 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 which the NTP server is willing and able to speak using the key material established through this NTS-KE 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.

4.1.3. Error

The Error record has a Record Type number of 2. Its body is exactly two octets long, consisting of an unsigned 16-bit integer in network byte order, denoting an error code. The Critical Bit MUST be set.

Clients MUST NOT include Error records in their request. If clients receive a server response which includes an Error record, they MUST discard any key material negotiated during the initial TLS exchange and MUST NOT proceed to the Next Protocol. Requirements for retry intervals are described in Section 4.2.

The following error codes are defined:

Error code 0 means "Unrecognized Critical Record". The server MUST respond with this error code if the request included a record which the server did not understand and which had its Critical Bit set. The client SHOULD NOT retry its request without modification.

Error code 1 means "Bad Request". The server MUST respond with this error if the request is not complete and syntactically well-formed, or, upon the expiration of an implementation-defined

timeout, it has not yet received such a request. The client SHOULD NOT retry its request without modification.

Error code 2 means "Internal Server Error". The server MUST respond with this error if it is unable to respond properly due to an internal condition. The client MAY retry its request.

4.1.4. Warning

The Warning record has a Record Type number of 3. Its body is exactly two octets long, consisting of an unsigned 16-bit integer in network byte order, denoting a warning code. The Critical Bit MUST be set.

Clients MUST NOT include Warning records in their request. If clients receive a server response which includes a Warning record, they MAY discard any negotiated key material and abort without proceeding to the Next Protocol. Unrecognized warning codes MUST be treated as errors.

This memo defines no warning codes.

4.1.5. AEAD Algorithm Negotiation

The AEAD Algorithm Negotiation record has a Record Type number of 4. Its body consists of a sequence of unsigned 16-bit integers in network byte order, denoting Numeric Identifiers from the IANA AEAD Algorithms registry [IANA-AEAD]. The Critical Bit MAY be set.

If the NTS Next Protocol Negotiation record offers Protocol ID 0 (for NTPv4), then this record MUST be included exactly once. Other protocols MAY require it as well.

When included in a request, this record denotes which AEAD algorithms the client is willing to use to secure the Next Protocol, in decreasing preference order. When included in a response, this record denotes which algorithm the server chooses to use. It is empty if the server supports none of the algorithms offered. In requests, the list MUST include at least one algorithm. In responses, it MUST include at most one. Honoring the client's preference order is OPTIONAL: servers may select among any of the client's offered choices, even if they are able to support some other algorithm which the client prefers more.

Server implementations of NTS extension fields for NTPv4 (Section 5) MUST support AEAD_AES_SIV_CMAC_256 [RFC5297] (Numeric Identifier 15). That is, if the client includes AEAD_AES_SIV_CMAC_256 in its AEAD Algorithm Negotiation record and the server accepts Protocol ID 0

(NTPv4) in its NTS Next Protocol Negotiation record, then the server's AEAD Algorithm Negotiation record MUST NOT be empty.

4.1.6. New Cookie for NTPv4

The New Cookie for NTPv4 record has a Record Type number of 5. The contents of its body SHALL be implementation-defined and clients MUST NOT attempt to interpret them. See Section 6 for a suggested construction.

Clients MUST NOT send records of this type. Servers MUST send at least one record of this type, and SHOULD send eight of them, if the Next Protocol Negotiation response record contains Protocol ID 0 (NTPv4) and the AEAD Algorithm Negotiation response record is not empty. The Critical Bit SHOULD NOT be set.

4.1.7. NTPv4 Server Negotiation

The NTPv4 Server Negotiation record has a Record Type number of 6. Its body consists of an ASCII-encoded [RFC0020] string. The contents of the string SHALL be either an IPv4 address, an IPv6 address, or a fully qualified domain name (FQDN). IPv4 addresses MUST be in dotted decimal notation. IPv6 addresses MUST conform to the "Text Representation of Addresses" as specified in RFC 4291 [RFC4291] and MUST NOT include zone identifiers [RFC6874]. If a label contains at least one non-ASCII character, it is an internationalized domain name and an A-LABEL MUST be used as defined in Section 2.3.2.1 of RFC 5890 [RFC5890]. If the record contains a domain name, the recipient MUST treat it as a FQDN, e.g. by making sure it ends with a dot.

When NTPv4 is negotiated as a Next Protocol and this record is sent by the server, the body specifies the hostname or IP address of the NTPv4 server with which the client should associate and which will accept the supplied cookies. If no record of this type is sent, the client SHALL interpret this as a directive to associate with an NTPv4 server at the same IP address as the NTS-KE server. Servers MUST NOT send more than one record of this type.

When this record is sent by the client, it indicates that the client wishes to associate with the specified NTP server. The NTS-KE server MAY incorporate this request when deciding what NTPv4 Server Negotiation records to respond with, but honoring the client's preference is OPTIONAL. The client MUST NOT send more than one record of this type.

If the client has sent a record of this type, the NTS-KE server SHOULD reply with the same record if it is valid and the server is able to supply cookies for it. If the client has not sent any record

of this type, the NTS-KE server SHOULD respond with either an NTP server address in the same family as the NTS-KE session or a FQDN that can be resolved to an address in that family, if such alternatives are available.

Servers MAY set the Critical Bit on records of this type; clients SHOULD NOT.

4.1.8. NTPv4 Port Negotiation

The NTPv4 Port Negotiation record has a Record Type number of 7. Its body consists of a 16-bit unsigned integer in network byte order, denoting a UDP port number.

When NTPv4 is negotiated as a Next Protocol and this record is sent by the server, the body specifies the port number of the NTPv4 server with which the client should associate and which will accept the supplied cookies. If no record of this type is sent, the client SHALL assume a default of 123 (the registered port number for NTP).

When this record is sent by the client in conjunction with a NTPv4 Server Negotiation record, it indicates that the client wishes to associate with the NTP server at the specified port. The NTS-KE server MAY incorporate this request when deciding what NTPv4 Server Negotiation and NTPv4 Port Negotiation records to respond with, but honoring the client's preference is OPTIONAL.

Servers MAY set the Critical Bit on records of this type; clients SHOULD NOT.

4.2. Retry Intervals

A mechanism for not unnecessarily overloading the NTS-KE server is REQUIRED when retrying the key establishment process due to protocol, communication, or other errors. The exact workings of this will be dependent on the application and operational experience gathered over time. Until such experience is available, this memo provides the following suggestion.

Clients SHOULD use exponential backoff, with an initial and minimum retry interval of 10 seconds, a maximum retry interval of 5 days, and a base of 1.5. Thus, the minimum interval in seconds, 't', for the nth retry is calculated with

$$t = \min(10 * 1.5^{(n-1)}, 432000).$$

Clients MUST NOT reset the retry interval until they have performed a successful key establishment with the NTS-KE server, followed by a

successful use of the negotiated next protocol with the keys and data established during that transaction.

4.3. Key Extraction (generally)

Following a successful run of the NTS-KE protocol, key material SHALL be extracted using the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) [RFC5869] in accordance with RFC 8446, Section 7.5 [RFC8446]. Inputs to the exporter function are to be constructed in a manner specific to the negotiated Next Protocol. However, all protocols which utilize NTS-KE MUST conform to the following two rules:

The disambiguating label string [RFC5705] MUST be "EXPORTER-network-time-security".

The per-association context value [RFC5705] MUST be provided and MUST begin with the two-octet Protocol ID which was negotiated as a Next Protocol.

5. NTS Extension Fields for NTPv4

5.1. Key Extraction (for NTPv4)

Following a successful run of the NTS-KE protocol wherein Protocol ID 0 (NTPv4) is selected as a Next Protocol, two AEAD keys SHALL be extracted: a client-to-server (C2S) key and a server-to-client (S2C) key. These keys SHALL be computed with the HKDF defined in RFC 8446, Section 7.5 [RFC8446] using the following inputs.

The disambiguating label string [RFC5705] SHALL be "EXPORTER-network-time-security".

The per-association context value [RFC5705] SHALL consist of the following five octets:

The first two octets SHALL be zero (the Protocol ID for NTPv4).

The next two octets SHALL be the Numeric Identifier of the negotiated AEAD Algorithm in network byte order.

The final octet SHALL be 0x00 for the C2S key and 0x01 for the S2C key.

Implementations wishing to derive additional keys for private or experimental use MUST NOT do so by extending the above-specified syntax for per-association context values. Instead, they SHOULD use their own disambiguating label string. Note that RFC 5705 [RFC5705]

provides that disambiguating label strings beginning with "EXPERIMENTAL" MAY be used without IANA registration.

5.2. Packet Structure Overview

In general, an NTS-protected NTPv4 packet consists of:

- The usual 48-octet NTP header which is authenticated but not encrypted.

- Some extension fields which are authenticated but not encrypted.

- An extension field which contains AEAD output (i.e., an authentication tag and possible ciphertext). The corresponding plaintext, if non-empty, consists of some extension fields which benefit from both encryption and authentication.

- Possibly, some additional extension fields which are neither encrypted nor authenticated. In general, these are discarded by the receiver.

Always included among the authenticated or authenticated-and-encrypted extension fields are a cookie extension field and a unique identifier extension field, as described in Section 5.7. The purpose of the cookie extension field is to enable the server to offload storage of session state onto the client. The purpose of the unique identifier extension field is to protect the client from replay attacks.

5.3. The Unique Identifier Extension Field

The Unique Identifier extension field provides the client with a cryptographically strong means of detecting replayed packets. It has a Field Type of [[TBD2]]. When the extension field is included in a client packet (mode 3), its body SHALL consist of a string of octets generated by a cryptographically secure random number generator [RFC4086]. The string MUST be at least 32 octets long. When the extension field is included in a server packet (mode 4), its body SHALL contain the same octet string as was provided in the client packet to which the server is responding. All server packets generated by NTS-implementing servers in response to client packets containing this extension field MUST also contain this field with the same content as in the client's request. The field's use in modes other than client-server is not defined.

This extension field MAY also be used standalone, without NTS, in which case it provides the client with a means of detecting spoofed packets from off-path attackers. Historically, NTP's origin

timestamp field has played both these roles, but for cryptographic purposes this is suboptimal because it is only 64 bits long and, depending on implementation details, most of those bits may be predictable. In contrast, the Unique Identifier extension field enables a degree of unpredictability and collision resistance more consistent with cryptographic best practice.

5.4. The NTS Cookie Extension Field

The NTS Cookie extension field has a Field Type of [[TBD3]]. Its purpose is to carry information which enables the server to recompute keys and other session state without having to store any per-client state. The contents of its body SHALL be implementation-defined and clients MUST NOT attempt to interpret them. See Section 6 for a suggested construction. The NTS Cookie extension field MUST NOT be included in NTP packets whose mode is other than 3 (client) or 4 (server).

5.5. The NTS Cookie Placeholder Extension Field

The NTS Cookie Placeholder extension field has a Field Type of [[TBD4]]. When this extension field is included in a client packet (mode 3), it communicates to the server that the client wishes it to send additional cookies in its response. This extension field MUST NOT be included in NTP packets whose mode is other than 3.

Whenever an NTS Cookie Placeholder extension field is present, it MUST be accompanied by an NTS Cookie extension field. The body length of the NTS Cookie Placeholder extension field MUST be the same as the body length of the NTS Cookie extension field. This length requirement serves to ensure that the response will not be larger than the request, in order to improve timekeeping precision and prevent DDoS amplification. The contents of the NTS Cookie Placeholder extension field's body SHOULD be all zeros and, aside from checking its length, MUST be ignored by the server.

5.6. The NTS Authenticator and Encrypted Extension Fields Extension Field

The NTS Authenticator and Encrypted Extension Fields extension field is the central cryptographic element of an NTS-protected NTP packet. Its Field Type is [[TBD5]]. It SHALL be formatted according to Figure 4 and include the following fields:

Nonce Length: Two octets in network byte order, giving the length of the Nonce field, excluding any padding, interpreted as an unsigned integer.

Ciphertext Length: Two octets in network byte order, giving the length of the Ciphertext field, excluding any padding, interpreted as an unsigned integer.

Nonce: A nonce as required by the negotiated AEAD Algorithm. The end of the field is zero-padded to a word (four octets) boundary.

Ciphertext: The output of the negotiated AEAD Algorithm. The structure of this field is determined by the negotiated algorithm, but it typically contains an authentication tag in addition to the actual ciphertext. The end of the field is zero-padded to a word (four octets) boundary.

Additional Padding: Clients which use a nonce length shorter than the maximum allowed by the negotiated AEAD algorithm may be required to include additional zero-padding. The necessary length of this field is specified below.

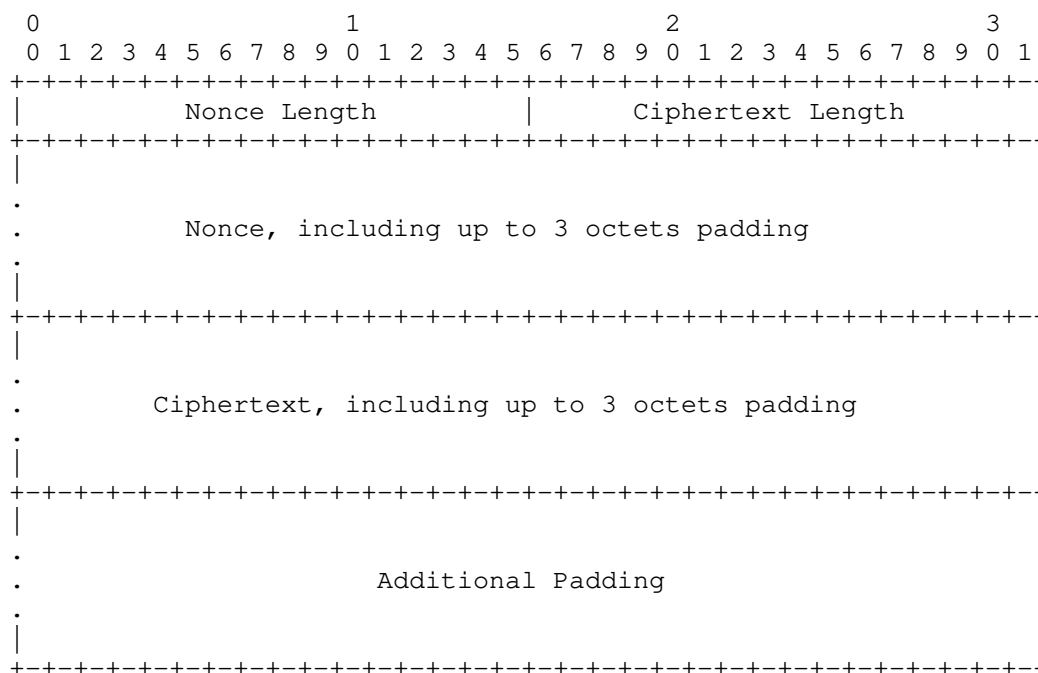


Figure 4: NTS Authenticator and Encrypted Extension Fields Extension Field Format

The Ciphertext field SHALL be formed by providing the following inputs to the negotiated AEAD Algorithm:

K: For packets sent from the client to the server, the C2S key SHALL be used. For packets sent from the server to the client, the S2C key SHALL be used.

A: The associated data SHALL consist of the portion of the NTP packet beginning from the start of the NTP header and ending at the end of the last extension field which precedes the NTS Authenticator and Encrypted Extension Fields extension field.

P: The plaintext SHALL consist of all (if any) NTP extension fields to be encrypted; if multiple extension fields are present they SHALL be joined by concatenation. Each such field SHALL be formatted in accordance with RFC 7822 [RFC7822], except that, contrary to the RFC 7822 requirement that fields have a minimum length of 16 or 28 octets, encrypted extension fields MAY be arbitrarily short (but still MUST be a multiple of 4 octets in length).

N: The nonce SHALL be formed however required by the negotiated AEAD algorithm.

The purpose of the Additional Padding field is to ensure that servers can always choose a nonce whose length is adequate to ensure its uniqueness, even if the client chooses a shorter one, and still ensure that the overall length of the server's response packet does not exceed the length of the request. For mode 4 (server) packets, no Additional Padding field is ever required. For mode 3 (client) packets, the length of the Additional Padding field SHALL be computed as follows. Let 'N_LEN' be the padded length of the Nonce field. Let 'N_MAX' be, as specified by RFC 5116 [RFC5116], the maximum permitted nonce length for the negotiated AEAD algorithm. Let 'N_REQ' be the lesser of 16 and N_MAX, rounded up to the nearest multiple of 4. If N_LEN is greater than or equal to N_REQ, then no Additional Padding field is required. Otherwise, the Additional Padding field SHALL be at least N_REQ - N_LEN octets in length. Servers MUST enforce this requirement by discarding any packet which does not conform to it.

Senders are always free to include more Additional Padding than mandated by the above paragraph. Theoretically, it could be necessary to do so in order to bring the extension field to the minimum length required by RFC 7822 [RFC7822]. This should never happen in practice because any reasonable AEAD algorithm will have a nonce and an authenticator long enough to bring the extension field to its required length already. Nonetheless, implementers are advised to explicitly handle this case and ensure that the extension field they emit is of legal length.

The NTS Authenticator and Encrypted Extension Fields extension field MUST NOT be included in NTP packets whose mode is other than 3 (client) or 4 (server).

5.7. Protocol Details

A client sending an NTS-protected request SHALL include the following extension fields as displayed in Figure 5:

Exactly one Unique Identifier extension field which MUST be authenticated, MUST NOT be encrypted, and whose contents MUST be the output of a cryptographically secure random number generator. [RFC4086]

Exactly one NTS Cookie extension field which MUST be authenticated and MUST NOT be encrypted. The cookie MUST be one which has been previously provided to the client, either from the key establishment server during the NTS-KE handshake or from the NTP server in response to a previous NTS-protected NTP request.

Exactly one NTS Authenticator and Encrypted Extension Fields extension field, generated using an AEAD Algorithm and C2S key established through NTS-KE.

To protect the client's privacy, the client SHOULD avoid reusing a cookie. If the client does not have any cookies that it has not already sent, it SHOULD initiate a re-run of the NTS-KE protocol. The client MAY reuse cookies in order to prioritize resilience over unlinkability. Which of the two that should be prioritized in any particular case is dependent on the application and the user's preference. Section 10.1 describes the privacy considerations of this in further detail.

The client MAY include one or more NTS Cookie Placeholder extension fields which MUST be authenticated and MAY be encrypted. The number of NTS Cookie Placeholder extension fields that the client includes SHOULD be such that if the client includes N placeholders and the server sends back N+1 cookies, the number of unused cookies stored by the client will come to eight. The client SHOULD NOT include more than seven NTS Cookie Placeholder extension fields in a request. When both the client and server adhere to all cookie-management guidance provided in this memo, the number of placeholder extension fields will equal the number of dropped packets since the last successful volley.

In rare circumstances, it may be necessary to include fewer NTS Cookie Placeholder extensions than recommended above in order to prevent datagram fragmentation. When cookies adhere the format

recommended in Section 6 and the AEAD in use is the mandatory-to-implement AEAD_AES_SIV_CMAC_256, senders can include a cookie and seven placeholders and still have packet size fall comfortably below 1280 octets if no non-NTS-related extensions are used; 1280 octets is the minimum prescribed MTU for IPv6 and is generally safe for avoiding IPv4 fragmentation. Nonetheless, senders SHOULD include fewer cookies and placeholders than otherwise indicated if doing so is necessary to prevent fragmentation.

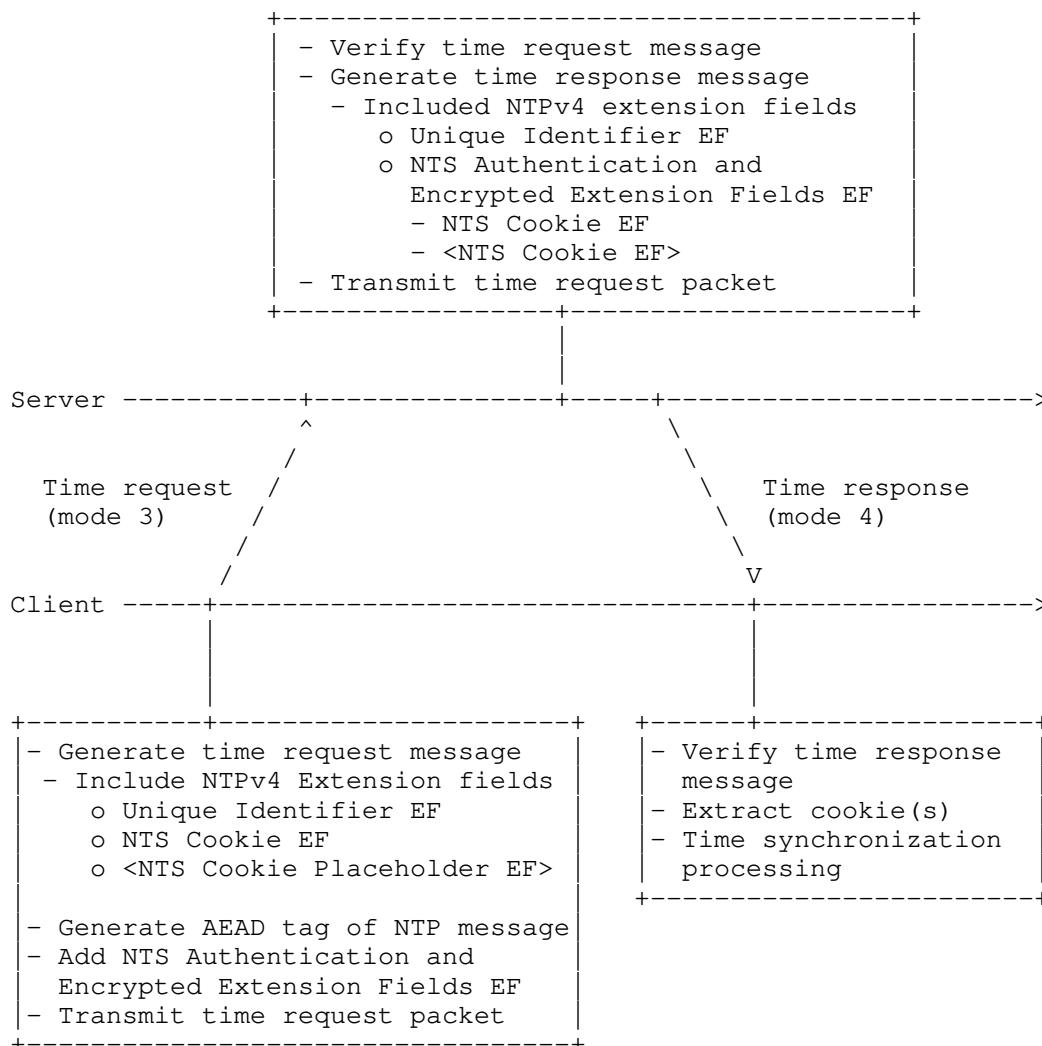


Figure 5: NTS-protected NTP Time Synchronization Messages

The client MAY include additional (non-NTS-related) extension fields which MAY appear prior to the NTS Authenticator and Encrypted Extension Fields extension fields (therefore authenticated but not encrypted), within it (therefore encrypted and authenticated), or after it (therefore neither encrypted nor authenticated). The server MUST discard any unauthenticated extension fields. Future specifications of extension fields MAY provide exceptions to this rule.

Upon receiving an NTS-protected request, the server SHALL (through some implementation-defined mechanism) use the cookie to recover the AEAD Algorithm, C2S key, and S2C key associated with the request, and then use the C2S key to authenticate the packet and decrypt the ciphertext. If the cookie is valid and authentication and decryption succeed, the server SHALL include the following extension fields in its response:

Exactly one Unique Identifier extension field which MUST be authenticated, MUST NOT be encrypted, and whose contents SHALL echo those provided by the client.

Exactly one NTS Authenticator and Encrypted Extension Fields extension field, generated using the AEAD algorithm and S2C key recovered from the cookie provided by the client.

One or more NTS Cookie extension fields which MUST be authenticated and encrypted. The number of NTS Cookie extension fields included SHOULD be equal to, and MUST NOT exceed, one plus the number of valid NTS Cookie Placeholder extension fields included in the request. The cookies returned in those fields MUST be valid for use with the NTP server that sent them. They MAY be valid for other NTP servers as well, but there is no way for the server to indicate this.

We emphasize the contrast that NTS Cookie extension fields MUST NOT be encrypted when sent from client to server, but MUST be encrypted when sent from server to client. The former is necessary in order for the server to be able to recover the C2S and S2C keys, while the latter is necessary to satisfy the unlinkability goals discussed in Section 10.1. We emphasize also that "encrypted" means encapsulated within the NTS Authenticator and Encrypted Extensions extension field. While the body of an NTS Cookie extension field will generally consist of some sort of AEAD output (regardless of whether the recommendations of Section 6 are precisely followed), this is not sufficient to make the extension field "encrypted".

The server MAY include additional (non-NTS-related) extension fields which MAY appear prior to the NTS Authenticator and Encrypted

Extension Fields extension field (therefore authenticated but not encrypted), within it (therefore encrypted and authenticated), or after it (therefore neither encrypted nor authenticated). The client MUST discard any unauthenticated extension fields. Future specifications of extension fields MAY provide exceptions to this rule.

Upon receiving an NTS-protected response, the client MUST verify that the Unique Identifier matches that of an outstanding request, and that the packet is authentic under the S2C key associated with that request. If either of these checks fails, the packet MUST be discarded without further processing. In particular, the client MUST discard unprotected responses to NTS-protected requests.

If the server is unable to validate the cookie or authenticate the request, it SHOULD respond with a Kiss-o'-Death (KoD) packet (see RFC 5905, Section 7.4 [RFC5905]) with kiss code "NTSN", meaning "NTS NAK" (NTS negative-acknowledgment). It MUST NOT include any NTS Cookie or NTS Authenticator and Encrypted Extension Fields extension fields.

If the NTP server has previously responded with authentic NTS-protected NTP packets, the client MUST verify that any KoD packets received from the server contain the Unique Identifier extension field and that the Unique Identifier matches that of an outstanding request. If this check fails, the packet MUST be discarded without further processing. If this check passes, the client MUST comply with RFC 5905, Section 7.4 [RFC5905] where required.

A client MAY automatically re-run the NTS-KE protocol upon forced disassociation from an NTP server. In that case, it MUST avoid quickly looping between the NTS-KE and NTP servers by rate limiting the retries. Requirements for retry intervals in NTS-KE are described in Section 4.2.

Upon reception of the NTS NAK kiss code, the client SHOULD wait until the next poll for a valid NTS-protected response and if none is received, initiate a fresh NTS-KE handshake to try to renegotiate new cookies, AEAD keys, and parameters. If the NTS-KE handshake succeeds, the client MUST discard all old cookies and parameters and use the new ones instead. As long as the NTS-KE handshake has not succeeded, the client SHOULD continue polling the NTP server using the cookies and parameters it has.

To allow for NTP session restart when the NTS-KE server is unavailable and to reduce NTS-KE server load, the client SHOULD keep at least one unused but recent cookie, AEAD keys, negotiated AEAD algorithm, and other necessary parameters on persistent storage.

This way, the client is able to resume the NTP session without performing renewed NTS-KE negotiation.

6. Suggested Format for NTS Cookies

This section is non-normative. It gives a suggested way for servers to construct NTS cookies. All normative requirements are stated in Section 4.1.6 and Section 5.4.

The role of cookies in NTS is closely analogous to that of session cookies in TLS. Accordingly, the thematic resemblance of this section to RFC 5077 [RFC5077] is deliberate and the reader should likewise take heed of its security considerations.

Servers should select an AEAD algorithm which they will use to encrypt and authenticate cookies. The chosen algorithm should be one such as AEAD_AES_SIV_CMAC_256 [RFC5297] which resists accidental nonce reuse. It need not be the same as the one that was negotiated with the client. Servers should randomly generate and store a secret master AEAD key 'K'. Servers should additionally choose a non-secret, unique value 'I' as key-identifier for 'K'.

Servers should periodically (e.g., once daily) generate a new pair '(I,K)' and immediately switch to using these values for all newly-generated cookies. Following each such key rotation, servers should securely erase any previously generated keys that should now be expired. Servers should continue to accept any cookie generated using keys that they have not yet erased, even if those keys are no longer current. Erasing old keys provides for forward secrecy, limiting the scope of what old information can be stolen if a master key is somehow compromised. Holding on to a limited number of old keys allows clients to seamlessly transition from one generation to the next without having to perform a new NTS-KE handshake.

The need to keep keys synchronized between NTS-KE and NTP servers as well as across load-balanced clusters can make automatic key rotation challenging. However, the task can be accomplished without the need for central key-management infrastructure by using a ratchet, i.e., making each new key a deterministic, cryptographically pseudo-random function of its predecessor. A recommended concrete implementation of this approach is to use HKDF [RFC5869] to derive new keys, using the key's predecessor as Input Keying Material and its key identifier as a salt.

To form a cookie, servers should first form a plaintext 'P' consisting of the following fields:

The AEAD algorithm negotiated during NTS-KE.

The S2C key.

The C2S key.

Servers should then generate a nonce 'N' uniformly at random, and form AEAD output 'C' by encrypting 'P' under key 'K' with nonce 'N' and no associated data.

The cookie should consist of the tuple '(I,N,C)'.

To verify and decrypt a cookie provided by the client, first parse it into its components 'I', 'N', and 'C'. Use 'I' to look up its decryption key 'K'. If the key whose identifier is 'I' has been erased or never existed, decryption fails; reply with an NTS NAK. Otherwise, attempt to decrypt and verify ciphertext 'C' using key 'K' and nonce 'N' with no associated data. If decryption or verification fails, reply with an NTS NAK. Otherwise, parse out the contents of the resulting plaintext 'P' to obtain the negotiated AEAD algorithm, S2C key, and C2S key.

7. IANA Considerations

7.1. 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: ntske

Transport Protocol: tcp

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Network Time Security Key Establishment

Reference: [[this memo]]

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

[[RFC EDITOR: Replace all instances of [[TBD1]] in this document with the IANA port assignment.]]

7.2. TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs Registry

IANA is requested to allocate the following entry in the TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs registry [RFC7301]:

Protocol: Network Time Security Key Establishment, version 1

Identification Sequence:

0x6E 0x74 0x73 0x6B 0x65 0x2F 0x31 ("ntske/1")

Reference: [[this memo]], Section 4

7.3. TLS Exporter Labels Registry

IANA is requested to allocate the following entry in the TLS Exporter Labels Registry [RFC5705]:

Value	DTLS-OK	Recommended	Reference	Note
EXPORTER-network-time-security	Y	Y	[[this memo]], Section 4.3	

7.4. NTP Kiss-o'-Death Codes Registry

IANA is requested to allocate the following entry in the registry of NTP Kiss-o'-Death Codes [RFC5905]:

Code	Meaning	Reference
NTSN	Network Time Security (NTS) negative-acknowledgment (NAK)	[[this memo]], Section 5.7

7.5. NTP Extension Field Types Registry

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

Field Type	Meaning	Reference
[[TBD2]]	Unique Identifier	[[this memo]], Section 5.3
[[TBD3]]	NTS Cookie	[[this memo]], Section 5.4
[[TBD4]]	NTS Cookie Placeholder	[[this memo]], Section 5.5
[[TBD5]]	NTS Authenticator and Encrypted Extension Fields	[[this memo]], Section 5.6

[[RFC EDITOR: REMOVE BEFORE PUBLICATION - The NTP WG suggests that the following values be used:

```
Unique Identifier      0x0104
NTS Cookie            0x0204
Cookie Placeholder    0x0304
NTS Authenticator     0x0404]]
```

[[RFC EDITOR: Replace all instances of [[TBD2]], [[TBD3]], [[TBD4]], and [[TBD5]] in this document with the respective IANA assignments.]]

7.6. Network Time Security Key Establishment Record Types Registry

IANA is requested to create a new registry entitled "Network Time Security Key Establishment Record Types". Entries SHALL have the following fields:

Record Type Number (REQUIRED): An integer in the range 0-32767 inclusive.

Description (REQUIRED): A short text description of the purpose of the field.

Reference (REQUIRED): A reference to a document specifying the semantics of the record.

The policy for allocation of new entries in this registry SHALL vary by the Record Type Number, as follows:

0-1023: IETF Review

1024-16383: Specification Required

16384-32767: Private and Experimental Use

The initial contents of this registry SHALL be as follows:

Record Type Number	Description	Reference
0	End of Message	[[this memo]], Section 4.1.1
1	NTS Next Protocol Negotiation	[[this memo]], Section 4.1.2
2	Error	[[this memo]], Section 4.1.3
3	Warning	[[this memo]], Section 4.1.4
4	AEAD Algorithm Negotiation	[[this memo]], Section 4.1.5
5	New Cookie for NTPv4	[[this memo]], Section 4.1.6
6	NTPv4 Server Negotiation	[[this memo]], Section 4.1.7
7	NTPv4 Port Negotiation	[[this memo]], Section 4.1.8
16384-32767	Reserved for Private & Experimental Use	[[this memo]]

7.7. Network Time Security Next Protocols Registry

IANA is requested to create a new registry entitled "Network Time Security Next Protocols". Entries SHALL have the following fields:

Protocol ID (REQUIRED): An integer in the range 0-65535 inclusive, functioning as an identifier.

Protocol Name (REQUIRED): A short text string naming the protocol being identified.

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

The policy for allocation of new entries in these registries SHALL vary by their Protocol ID, as follows:

0-1023: IETF Review

1024-32767: Specification Required

32768-65535: Private and Experimental Use

The initial contents of this registry SHALL be as follows:

Protocol ID	Protocol Name	Reference
0	Network Time Protocol version 4 (NTPv4)	[[this memo]]
32768-65535	Reserved for Private or Experimental Use	Reserved by [[this memo]]

7.8. Network Time Security Error and Warning Codes Registries

IANA is requested to create two new registries entitled "Network Time Security Error Codes" and "Network Time Security Warning Codes".

Entries in each SHALL have the following fields:

Number (REQUIRED): An integer in the range 0-65535 inclusive

Description (REQUIRED): A short text description of the condition.

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

The policy for allocation of new entries in these registries SHALL vary by their Number, as follows:

0-1023: IETF Review

1024-32767: Specification Required

32768-65535: Private and Experimental Use

The initial contents of the Network Time Security Error Codes Registry SHALL be as follows:

Number	Description	Reference
0	Unrecognized Critical Extension	[[this memo]], Section 4.1.3
1	Bad Request	[[this memo]], Section 4.1.3
2	Internal Server Error	[[this memo]], Section 4.1.3
32768-65535	Reserved for Private or Experimental Use	Reserved by [[this memo]]

The Network Time Security Warning Codes Registry SHALL initially be empty except for the reserved range, i.e.:

Number	Description	Reference
32768-65535	Reserved for Private or Experimental Use	Reserved by [[this memo]]

8. Implementation Status - RFC EDITOR: REMOVE BEFORE PUBLICATION

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

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

8.1. Implementation 1

Organization: Ostfalia University of Applied Science

Implementor: Martin Langer

Maturity: Proof-of-Concept Prototype

This implementation was used to verify consistency and to ensure completeness of this specification.

8.1.1. Coverage

This implementation covers the complete specification.

8.1.2. Licensing

The code is released under a Apache License 2.0 license.

The source code is available at: <https://gitlab.com/MLanger/nts/>

8.1.3. Contact Information

Contact Martin Langer: mart.langer@ostfalia.de

8.1.4. Last Update

The implementation was updated 25. February 2019.

8.2. Implementation 2

Organization: Netnod

Implementor: Christer Weinigel

Maturity: Proof-of-Concept Prototype

This implementation was used to verify consistency and to ensure completeness of this specification.

8.2.1. Coverage

This implementation covers the complete specification.

8.2.2. Licensing

The source code is available at: <https://github.com/Netnod/nts-poc-python>.

See LICENSE file for details on licensing (BSD 2).

8.2.3. Contact Information

Contact Christer Weinigel: christer@weinigel.se

8.2.4. Last Update

The implementation was updated 31. January 2019.

8.3. Implementation 3

Organization: Red Hat

Implementor: Miroslav Lichvar

Maturity: Prototype

This implementation was used to verify consistency and to ensure completeness of this specification.

8.3.1. Coverage

This implementation covers the complete specification.

8.3.2. Licensing

Licensing is GPLv2.

The source code is available at: <https://github.com/mlichvar/chrony-nts>

8.3.3. Contact Information

Contact Miroslav Lichvar: mlichvar@redhat.com

8.3.4. Last Update

The implementation was updated 28. March 2019.

8.4. Implementation 4

Organization: NTPsec

Implementor: Hal Murray and NTPsec team

Maturity: Looking for testers. Servers running at ntp1.glypnod.com:123 and ntp2.glypnod.com:123

This implementation was used to verify consistency and to ensure completeness of this specification.

8.4.1. Coverage

This implementation covers the complete specification.

8.4.2. Licensing

The source code is available at: <https://gitlab.com/NTPsec/ntpsec>.
Licensing details in LICENSE.

8.4.3. Contact Information

Contact Hal Murray: hmurray@megapathdsl.net, devel@ntpsec.org

8.4.4. Last Update

The implementation was updated 2019-Apr-10.

8.5. Implementation 5

Organization: Cloudflare

Implementor: Watson Ladd

Maturity:

This implementation was used to verify consistency and to ensure completeness of this specification.

8.5.1. Coverage

This implementation covers the server side of the NTS specification.

8.5.2. Licensing

The source code is available at: <https://github.com/wbl/nts-rust>

Licensing is ISC (details see LICENSE.txt file).

8.5.3. Contact Information

Contact Watson Ladd: watson@cloudflare.com

8.5.4. Last Update

The implementation was updated 21. March 2019.

8.6. Implementation 6

Organization: Hacklunch, independent

Implementor: Michael Cardell Widerkrantz, Daniel Lublin, Martin Samuelsson et. al.

Maturity: interoperable client, immature server

8.6.1. Coverage

NTS-KE client and server.

8.6.2. Licensing

Licensing is ISC (details in LICENSE file).

Source code is available at: <https://gitlab.com/hacklunch/ntsclient>

8.6.3. Contact Information

Contact Michael Cardell Widerkrantz: mc@netnod.se

8.6.4. Last Update

The implementation was updated 6. February 2020.

8.7. Interoperability

The Interoperability tests distinguished between NTS key establishment protocol and NTS time exchange messages. For the implementations 1, 2, 3, and 4 pairwise interoperability of the NTS key establishment protocol and exchange of NTS protected NTP messages have been verified successfully. The implementation 2 was able to successfully perform the key establishment protocol against the server side of the implementation 5.

These tests successfully demonstrate that there are at least four running implementations of this draft which are able to interoperate.

9. Security Considerations

9.1. Protected Modes

NTP provides many different operating modes in order to support different network topologies and to adapt to various requirements. This memo only specifies NTS for NTP modes 3 (client) and 4 (server) (see Section 1.2). The best current practice for authenticating the other NTP modes is using the symmetric message authentication code feature as described in RFC 5905 [RFC5905] and RFC 8573 [RFC8573].

9.2. Cookie Encryption Key Compromise

If the suggested format for NTS cookies in Section 6 of this draft is used, an attacker who has gained access to the secret cookie encryption key 'K' can impersonate the NTP server, including generating new cookies. NTP and NTS-KE server operators SHOULD remove compromised keys as soon as the compromise is discovered. This will cause the NTP servers to respond with NTS NAK, thus forcing key renegotiation. Note that this measure does not protect against MITM attacks where the attacker has access to a compromised cookie encryption key. If another cookie scheme is used, there are likely similar considerations for that particular scheme.

9.3. Sensitivity to DDoS Attacks

The introduction of NTS brings with it the introduction of asymmetric cryptography to NTP. Asymmetric cryptography is necessary for initial server authentication and AEAD key extraction. Asymmetric cryptosystems are generally orders of magnitude slower than their symmetric counterparts. This makes it much harder to build systems that can serve requests at a rate corresponding to the full line speed of the network connection. This, in turn, opens up a new possibility for DDoS attacks on NTP services.

The main protection against these attacks in NTS lies in that the use of asymmetric cryptosystems is only necessary in the initial NTS-KE phase of the protocol. Since the protocol design enables separation of the NTS-KE and NTP servers, a successful DDoS attack on an NTS-KE server separated from the NTP service it supports will not affect NTP users that have already performed initial authentication, AEAD key extraction, and cookie exchange.

NTS users should also consider that they are not fully protected against DoS attacks by on-path adversaries. In addition to dropping packets and attacks such as those described in Section 9.6, an on-path attacker can send spoofed kiss-o'-death replies, which are not authenticated, in response to NTP requests. This could result in significantly increased load on the NTS-KE server. Implementers have to weigh the user's need for unlinkability against the added resilience that comes with cookie reuse in cases of NTS-KE server unavailability.

9.4. Avoiding DDoS Amplification

Certain non-standard and/or deprecated features of the Network Time Protocol enable clients to send a request to a server which causes the server to send a response much larger than the request. Servers which enable these features can be abused in order to amplify traffic

volume in DDoS attacks by sending them a request with a spoofed source IP. In recent years, attacks of this nature have become an endemic nuisance.

NTS is designed to avoid contributing any further to this problem by ensuring that NTS-related extension fields included in server responses will be the same size as the NTS-related extension fields sent by the client. In particular, this is why the client is required to send a separate and appropriately padded-out NTS Cookie Placeholder extension field for every cookie it wants to get back, rather than being permitted simply to specify a desired quantity.

Due to the RFC 7822 [RFC7822] requirement that extensions be padded and aligned to four-octet boundaries, response size may still in some cases exceed request size by up to three octets. This is sufficiently inconsequential that we have declined to address it.

9.5. Initial Verification of Server Certificates

NTS's security goals are undermined if the client fails to verify that the X.509 certificate chain presented by the NTS-KE server is valid and rooted in a trusted certificate authority. RFC 5280 [RFC5280] and RFC 6125 [RFC6125] specify how such verification is to be performed in general. However, the expectation that the client does not yet have a correctly-set system clock at the time of certificate verification presents difficulties with verifying that the certificate is within its validity period, i.e., that the current time lies between the times specified in the certificate's notBefore and notAfter fields. It may be operationally necessary in some cases for a client to accept a certificate which appears to be expired or not yet valid. While there is no perfect solution to this problem, there are several mitigations the client can implement to make it more difficult for an adversary to successfully present an expired certificate:

- Check whether the system time is in fact unreliable. On systems with the `ntp_adjtime()` system call, a return code other than `TIME_ERROR` indicates that some trusted software has already set the time and certificates can be strictly validated.

- Allow the system administrator to specify that certificates should **always** be strictly validated. Such a configuration is appropriate on systems which have a battery-backed clock and which can reasonably prompt the user to manually set an approximately-correct time if it appears to be needed.

- Once the clock has been synchronized, periodically write the current system time to persistent storage. Do not accept any

certificate whose notAfter field is earlier than the last recorded time.

NTP time replies are expected to be consistent with the NTS-KE TLS certificate validity period, i.e. time replies received immediately after an NTS-KE handshake are expected to lie within the certificate validity period. Implementations are recommended to check that this is the case. Performing a new NTS-KE handshake based solely on the fact that the certificate used by the NTS-KE server in a previous handshake has expired is normally not necessary. Clients that still wish to do this must take care not to cause an inadvertent denial-of-service attack on the NTS-KE server, for example by picking a random time in the week preceding certificate expiry to perform the new handshake.

Use multiple time sources. The ability to pass off an expired certificate is only useful to an adversary who has compromised the corresponding private key. If the adversary has compromised only a minority of servers, NTP's selection algorithm (RFC 5905 section 11.2.1 [RFC5905]) will protect the client from accepting bad time from the adversary-controlled servers.

9.6. Delay Attacks

In a packet delay attack, an adversary with the ability to act as a man-in-the-middle delays time synchronization packets between client and server asymmetrically [RFC7384]. Since NTP's formula for computing time offset relies on the assumption that network latency is roughly symmetrical, this leads to the client to compute an inaccurate value [Mizrahi]. The delay attack does not reorder or modify the content of the exchanged synchronization packets. Therefore, cryptographic means do not provide a feasible way to mitigate this attack. However, the maximum error that an adversary can introduce is bounded by half of the round trip delay.

RFC 5905 [RFC5905] specifies a parameter called MAXDIST which denotes the maximum round-trip latency (including not only the immediate round trip between client and server, but the whole distance back to the reference clock as reported in the Root Delay field) that a client will tolerate before concluding that the server is unsuitable for synchronization. The standard value for MAXDIST is one second, although some implementations use larger values. Whatever value a client chooses, the maximum error which can be introduced by a delay attack is MAXDIST/2.

Usage of multiple time sources, or multiple network paths to a given time source [Shpiner], may also serve to mitigate delay attacks if the adversary is in control of only some of the paths.

9.7. NTS Stripping

Implementers must be aware of the possibility of "NTS stripping" attacks, where an attacker attempts to trick clients into reverting to plain NTP. Naive client implementations might, for example, revert automatically to plain NTP if the NTS-KE handshake fails. A man-in-the-middle attacker can easily cause this to happen. Even clients that already hold valid cookies can be vulnerable, since an attacker can force a client to repeat the NTS-KE handshake by sending faked NTP mode 4 replies with the NTS NAK kiss code. Forcing a client to repeat the NTS-KE handshake can also be the first step in more advanced attacks.

For the reasons described here, implementations SHOULD NOT revert from NTS-protected to unprotected NTP with any server without explicit user action.

10. Privacy Considerations

10.1. Unlinkability

Unlinkability prevents a device from being tracked when it changes network addresses (e.g. because said device moved between different networks). In other words, unlinkability thwarts an attacker that seeks to link a new network address used by a device with a network address that it was formerly using, because of recognizable data that the device persistently sends as part of an NTS-secured NTP association. This is the justification for continually supplying the client with fresh cookies, so that a cookie never represents recognizable data in the sense outlined above.

NTS's unlinkability objective is merely to not leak any additional data that could be used to link a device's network address. NTS does not rectify legacy linkability issues that are already present in NTP. Thus, a client that requires unlinkability must also minimize information transmitted in a client query (mode 3) packet as described in the draft [I-D.ietf-ntp-data-minimization].

The unlinkability objective only holds for time synchronization traffic, as opposed to key establishment traffic. This implies that it cannot be guaranteed for devices that function not only as time clients, but also as time servers (because the latter can be externally triggered to send linkable data, such as the TLS certificate).

It should also be noted that it could be possible to link devices that operate as time servers from their time synchronization traffic, using information exposed in (mode 4) server response packets (e.g.

reference ID, reference time, stratum, poll). Also, devices that respond to NTP control queries could be linked using the information revealed by control queries.

Note that the unlinkability objective does not prevent a client device to be tracked by its time servers.

10.2. Confidentiality

NTS does not protect the confidentiality of information in NTP's header fields. When clients implement [I-D.ietf-ntp-data-minimization], client packet headers do not contain any information which the client could conceivably wish to keep secret: one field is random, and all others are fixed. Information in server packet headers is likewise public: the origin timestamp is copied from the client's (random) transmit timestamp, and all other fields are set the same regardless of the identity of the client making the request.

Future extension fields could hypothetically contain sensitive information, in which case NTS provides a mechanism for encrypting them.

11. Acknowledgements

The authors would like to thank Richard Barnes, Steven Bellovin, Scott Fluhrer, Patrik Faltstroom (Faltstrom), Sharon Goldberg, Russ Housley, Benjamin Kaduk, Suresh Krishnan, Mirja Kuehlewind (Kuehlewind), Martin Langer, Barry Leiba, Miroslav Lichvar, Aanchal Malhotra, Danny Mayer, Dave Mills, Sandra Murphy, Hal Murray, Karen O'Donoghue, Eric K. Rescorla, Kurt Roeckx, Stephen Roettger, Dan Romascanu, Kyle Rose, Rich Salz, Brian Sniffen, Susan Sons, Douglas Stebila, Harlan Stenn, Joachim Strombergsson (Strombergsson), Martin Thomson, Eric (Eric) Vyncke, Richard Welty, Christer Weinigel, and Magnus Westerlund for contributions to this document and comments on the design of NTS.

12. References

12.1. Normative References

[IANA-AEAD]

IANA, "Authenticated Encryption with Associated Data (AEAD) Parameters",
<<https://www.iana.org/assignments/aead-parameters/>>.

- [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>>.
- [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>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [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>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [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>>.
- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <<https://www.rfc-editor.org/info/rfc5705>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [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>>.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [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>>.
- [RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, DOI 10.17487/RFC6874, February 2013, <<https://www.rfc-editor.org/info/rfc6874>>.
- [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>>.
- [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>>.
- [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>>.

12.2. Informative 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.

- [Mizrahi] Mizrahi, T., "A game theoretic analysis of delay attacks against time synchronization protocols", in Proceedings of Precision Clock Synchronization for Measurement Control and Communication, ISPCS 2012, pp. 1-6, DOI 10.1109/ISPCS.2012.6336612, September 2012.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <<https://www.rfc-editor.org/info/rfc5077>>.
- [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>>.
- [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>>.
- [Shpiner] Shpiner, A., Revah, Y., and T. Mizrahi, "Multi-path Time Protocols", in Proceedings of IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS), DOI 10.1109/ISPCS.2013.6644754, September 2013.

Appendix A. Terms and Abbreviations

AEAD	Authenticated Encryption with Associated Data [RFC5116]
ALPN	Application-Layer Protocol Negotiation [RFC7301]
C2S	Client-to-server
DoS	Denial-of-Service
DDoS	Distributed Denial-of-Service
EF	Extension Field [RFC5905]
HKDF	Hashed Message Authentication Code-based Key Derivation Function [RFC5869]

KoD Kiss-o'-Death [RFC5905]
NTP Network Time Protocol [RFC5905]
NTS Network Time Security
NTS NAK NTS negative-acknowledgment
NTS-KE Network Time Security Key Establishment
S2C Server-to-client
TLS Transport Layer Security [RFC8446]

Authors' Addresses

Daniel Fox Franke
Akamai Technologies
145 Broadway
Cambridge, MA 02142
United States

Email: dafranke@akamai.com

Dieter Sibold
Physikalisch-Technische
Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8420
Fax: +49-531-592-698420
Email: dieter.sibold@ptb.de

Kristof Teichel
Physikalisch-Technische
Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-4471
Email: kristof.teichel@ptb.de

Marcus Dansarie
Sweden

Email: marcus@dansarie.se

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

Ragnar Sundblad
Netnod
Sweden

Email: ragge@netnod.se

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 September 2022

N. Wu
D. Dhody, Ed.
Huawei
A. Sinha, Ed.
A. Kumar S N
RtBrick Inc.
Y. Zhao
Ericsson
20 March 2022

A YANG Data Model for NTP
draft-ietf-ntp-yang-data-model-17

Abstract

This document defines a YANG data model for Network Time Protocol (NTP) version 4 implementations. It can also be used to configure version 3. 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 21 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Operational State	3
1.2. Terminology	3
1.3. Tree Diagrams	3
1.4. Prefixes in Data Node Names	3
1.5. References in the Model	4
2. NTP data model	5
3. Relationship with NTPv4-MIB	7
4. Relationship with RFC 7317	9
5. Access Rules	9
6. Key Management	10
7. NTP Version	10
8. NTP YANG Module	11
9. Usage Example	41
9.1. Unicast association	41
9.2. Refclock master	44
9.3. Authentication configuration	44
9.4. Access configuration	45
9.5. Multicast configuration	46
9.6. Manycast configuration	50
9.7. Clock state	53
9.8. Get all association	53
9.9. Global statistic	55
10. IANA Considerations	55
10.1. IETF XML Registry	55
10.2. YANG Module Names	55
11. Security Considerations	56
12. Acknowledgments	57
13. References	58
13.1. Normative References	58
13.2. Informative References	59
Appendix A. Full YANG Tree	60

Authors' Addresses	64
------------------------------	----

1. Introduction

This document defines a YANG [RFC7950] data model for Network Time Protocol [RFC5905] implementations. Note that the model could also be used to configure NTPv3 [RFC1305] (see Section 7).

The data model covers configuration of system parameters of NTP, such as access rules, authentication and VPN Routing and Forwarding (VRF) binding, and also various modes of NTP and per-interface parameters. It also provides access to 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] and [RFC1305].

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]
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]
FIPS 180-4: Secure Hash Standard (SHS)	[SHS]

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
|   |   +--...
|   +--rw associations
|   |   +--ro association* [address local-mode isconfigured]
|   |   |   +--ro address        inet:ip-address
|   |   |   +--ro local-mode     identityref
|   |   |   +--ro isconfigured   boolean
|   |   |   +--...
|   |   +--ro ntp-statistics
|   |   |   +--...
|   +--rw interfaces
|   |   +--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
+---...

rpcs:
+---x statistics-reset
+---w input
+---w (association-or-all)?
+--:(association)
|   +---w associations-address?
|   |   -> /ntp/associations/association/address
|   +---w associations-local-mode?
|   |   -> /ntp/associations/association/local-mode
|   +---w associations-isconfigured?
|   |   -> /ntp/associations/association/isconfigured
+--:(all)

```

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

Table 3

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

Table 4

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

Table 5

YANG NTP Configuration Data Nodes and counterparts in RFC 7317 Objects

5. Access Rules

The access rules in this section refers to the on-the-wire access control to the NTP service and completely independent of any management API access control, e.g., NETCONF Access Control Model (NACM) ([RFC8341]).

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] (for NTPv3) and [RFC5905] (for NTPv4), 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] (for NTPv3) and [RFC5905] (for NTPv4), 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.

The 'nacm:default-deny-all' is used to prevent retrieval of the actual key information after it is set.

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. Note that the version 3 of NTP [RFC1305] is obsoleted by NTPv4 [RFC5905].

8. NTP YANG Module

```
<CODE BEGINS> file "ietf-ntp@2022-03-21.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:  <https://datatracker.ietf.org/wg/ntp/about/>
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.
```

```
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.
```

```
Copyright (c) 2022 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 Revised 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.";
```

```
revision 2022-03-21 {
```

```
  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 {
    range "3..max";
  }
  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 date-and-time format when valid value exist,
        otherwise allows for setting special value such as
        zero.";
    reference
        "RFC 6991: Common YANG Data Types";
}

typedef log2seconds {
    type int8;
    description
        "An 8-bit signed integer that represents signed log2
        seconds.";
}

/* 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 mechanism.
        It is enabled to support legacy compatibility when secure
        cryptographic algorithms are not available to use.
        It is also used to configure keystings in ASCII format.";
    reference
        "RFC 1321: The MD5 Message-Digest Algorithm
        RFC 3174: US Secure Hash Algorithm 1 (SHA1)"
}
```

```
        FIPS 180-4: Secure Hash Standard (SHS)";
    }

    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 where the unicast server
      address is configured.";
}

identity uc-peer {
  if-feature "unicast-configuration";
  base unicast-configuration-type;
  description
    "Use symmetric active association mode where the peer
      address is configured.";
```

```
    }

    /* 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 identify
            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-never-set {
        base ntp-sync-state;
        description
            "Indicates the clock was never set.";
    }
```

```
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 md5 {
  if-feature "deprecated";
  base crypto-algorithm;
  description
    "The MD5 algorithm. Note that RFC 8573
    deprecates the use of MD5-based authentication.";
  reference
    "RFC 1321: The MD5 Message-Digest Algorithm";
}

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

```
    "RFC 3174: US Secure Hash Algorithm 1 (SHA1)";
  }

  identity hmac-sha-1 {
    if-feature "deprecated";
    base crypto-algorithm;
    description
      "HMAC-SHA-1 authentication algorithm.";
    reference
      "FIPS 180-4: Secure Hash Standard (SHS)";
  }

  identity hmac-sha1-12 {
    if-feature "deprecated";
    base crypto-algorithm;
    description
      "The HMAC-SHA1-12 algorithm.";
  }

  identity hmac-sha-256 {
    description
      "HMAC-SHA-256 authentication algorithm.";
    reference
      "FIPS 180-4: Secure Hash Standard (SHS)";
  }

  identity hmac-sha-384 {
    description
      "HMAC-SHA-384 authentication algorithm.";
    reference
      "FIPS 180-4: Secure Hash Standard (SHS)";
  }

  identity hmac-sha-512 {
    description
      "HMAC-SHA-512 authentication algorithm.";
    reference
      "FIPS 180-4: Secure Hash Standard (SHS)";
  }

  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
       RFC 8573: Message Authentication Code for the Network
```

```
        Time Protocol";
    }

    /* Groupings */

    grouping key {
        description
            "The key.";
        nacm:default-deny-all;
        choice key-string-style {
            description
                "Key string styles";
            case keystack {
                leaf keystack {
                    if-feature "deprecated";
                    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.";
        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 and 7.4";
}

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 log2seconds;
    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 log2seconds;
    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 | 1024..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:association"
        + "/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:association"
                + "/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:association/"
                + "ntp:isconfigured";
        }
        description
            "Indicates if the association (that resulted in the
            clock synchronization) is explicitly configured.";
    }
}

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 | 1024..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
        "The NTP access mode. Some of the possible value
         includes peer, server, synchronization, query
         etc.";
    }
    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. Some of the possible value
        includes synchronized and unsynchronized";
    }
    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 unsynchronized.";
      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 log2seconds;
  mandatory true;
  description
    "Clock precision of this system in signed integer format,
    in log 2 seconds - (prec=2^(-n)). A value of 5 would
    mean 2^-5 = 0.03125 seconds = 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 signed time offset to the current selected reference
       time source e.g., '0.032ms' or '1.232ms'. The negative
       value Indicates that the local clock is behind the
       current selected reference time source.";
    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, 7.3 and 10.";
  }
  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
            "The unicast configuration type, for example
            unicast-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.";
    }
  }
  container associations {
    description
      "Association parameters";
    list association {
      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 remote 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 (true) or
       dynamically learned (false).";
  }
  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
      "It is an 8-bit shift register that tracks packet
      generation and receipt. It is used to determine
      whether the server is reachable and the data are
      fresh.";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
      Algorithms Specification, Section 9.2 and 13";
  }
  leaf unreachable {
    type uint8;
    units "seconds";
    description
      "It is a count of how long in second the server has been
      unreachable i.e. the reach value has been zero.";
    reference
      "RFC 5905: Network Time Protocol Version 4: Protocol and
      Algorithms Specification, Section 9.2 and 13";
  }
  leaf poll {
    type log2seconds;
    description
      "The polling interval for current association in signed
```



```
        log2 seconds.";
    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 signed offset between the local clock
        and the peer clock, e.g., '0.032ms' or '1.232ms'. The
        negative value Indicates that the local clock is behind
        the peer.";
    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 on this
        interface";
      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 on this interface.";
        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 on this
        interface.";
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 on this interface.";
        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 on this interface.";
  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 anycast 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 anycast candidates in this
            association.";
        reference
            "RFC 5905: Network Time Protocol Version 4: Protocol and
            Algorithms Specification, Section 13.2";
    }
    leaf beacon {
        type log2seconds;
        description
            "The beacon is the upper limit of poll interval. When the
            ttl reaches its limit without finding the minimum number
            of anycast 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.";
      }
  }
}

rpc statistics-reset {
  description
    "Reset statistics collected.";
  input {
    choice association-or-all {
      description
        "Resets statistics for a particular association or
         all";
      case association {
        uses association-ref;
        description
          "This resets all the statistics collected for
           the association.";
      }
      case all {
        description
          "This resets all the statistics collected.";
      }
    }
  }
}
}
<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 use 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>
```

```
        </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 -

```
<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>
```

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. Multicast configuration

This example describes how to configure multicast-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>
          <multicast-client>
            <address>224.0.1.1</address>
            <authentication>
              <symmetric-key>
                <key-id>10</key-id>
              </symmetric-key>
            </authentication>
            <port>1025</port>
          </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>
          <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>
          <multicast-server>
            <address>224.0.1.1</address>
          </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>
        </multicast-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>
      <association>
        <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>
      </association>
    </associations>
  </ntp>
</data>
```

```
    </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

10.1. IETF XML Registry

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.

10.2. YANG Module Names

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. The 'nacm:default-deny-all' is used to prevent retrieval of the key information.

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. It could also lead to synchronization over untrusted source over trusted ones.

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. Unauthorized access to the keys can be easily exploited to permit unauthorized access to the NTP service. This information is sensitive and thus unauthorized access to this needs to be curtailed.

/ntp/associations/association/ - The entries in the list includes all active NTP associations of all modes. Exposure of these nodes could reveal network topology or trust relationship. Unauthorized access to this also needs to be curtailed.

/ntp/authentication and /ntp/access-rules - The entries in the list include the authentication and access control configurations. Exposure of these nodes could reveal network topology or trust relationship.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

statistics-reset - The RPC is used to reset statistics. Unauthorized reset could impact monitoring.

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.

Thanks for the IESG review from Benjamin Kaduk, Francesca Palombini, Eric Vyncke, Murray Kucherawy, Robert Wilton, Roman Danyliw, and Zaheduzzaman Sarker.

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

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [SHS] NIST, "Secure Hash Standard (SHS)", FIPS PUB 180-4, March 2012, <<https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf>>.

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 {deprecated}?
    |     |   +--:(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/association/address
    | +--ro associations-local-mode?
    |   | -> /ntp/associations/association/local-mode
    | +--ro associations-isconfigured?
    |   | -> /ntp/associations/association/isconfigured
    | +--ro nominal-freq      decimal64
    | +--ro actual-freq       decimal64
    | +--ro clock-precision   log2seconds
    | +--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?               log2seconds
+---rw maxpoll?               log2seconds
+---rw port?                  inet:port-number {ntp-port}?
+---rw version?               ntp-version
+---rw associations
+---ro association* [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?               log2seconds
+---ro maxpoll?               log2seconds
+---ro port?                  inet:port-number {ntp-port}?
+---ro version?               ntp-version
+---ro reach?                  uint8
+---ro unreach?               uint8
+---ro poll?                   log2seconds
+---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?      log2seconds
    +---rw maxpoll?      log2seconds
    +---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?      log2seconds
|   |   +---rw maxpoll?      log2seconds
|   |   +---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?       log2seconds
|   +---rw minpoll?      log2seconds
|   +---rw maxpoll?      log2seconds
|   +---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

rpcs:
+---x statistics-reset
+---w input

```

```
+---w (association-or-all)?
+--:(association)
|   +---w associations-address?
|   |       -> /ntp/associations/association/address
|   +---w associations-local-mode?
|   |       -> /ntp/associations/association/local-mode
|   +---w associations-isconfigured?
|   |       -> /ntp/associations/association/isconfigured
+--:(all)
```

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 560066
Karnataka
India
Email: dhruv.ietf@gmail.com

Ankit kumar Sinha (editor)
RtBrick Inc.
Bangalore
Karnataka
India
Email: ankit.ietf@gmail.com

Anil Kumar S N
RtBrick Inc.
Bangalore
Karnataka
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

Internet Working Group

Internet-Draft

Intended status: Standards Track

Expires: July 2019

Y. Jiang, Ed.
Huawei
X. Liu
Independent
J. Xu
Huawei
R. Cummings, Ed.
National Instruments
January 3, 2019

YANG Data Model for IEEE 1588-2008
draft-ietf-tictoc-1588v2-yang-11

Abstract

This document defines a YANG data model for the configuration of IEEE 1588-2008 devices and clocks, and also retrieval of the configuration information, data set and running states of IEEE 1588-2008 clocks. The YANG module in this document conforms to the Network Management Datastore Architecture (NMDA).

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on July 3, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction	2
1.1.	Conventions used in this document	4
1.2.	Terminology	4
2.	IEEE 1588-2008 YANG Model hierarchy	5
2.1.	Interpretations from IEEE 1588 Working Group	8
2.2.	Configuration and state	8
3.	IEEE 1588-2008 YANG Module	9
4.	Security Considerations	22
5.	IANA Considerations	23
6.	References	23
6.1.	Normative References	23
6.2.	Informative References	24
7.	Acknowledgments	25
	Appendix A Transferring YANG Work to IEEE 1588 WG	26
	A.1. Assumptions for the Transfer	27
	A.2. Intellectual Property Considerations	27
	A.3. Namespace and Module Name	28
	A.4. IEEE 1588 YANG Modules in ASCII Format	29

1. Introduction

As a synchronization protocol, IEEE 1588-2008 [IEEE1588] is widely supported in the carrier networks, industrial networks, automotive networks, and many other applications. It can provide high precision time synchronization as fine as nano-seconds. The protocol depends on a Precision Time Protocol (PTP) engine to decide its own state automatically, and a PTP transportation layer to carry the PTP timing and various quality messages. The

configuration parameters and state data sets of IEEE 1588-2008 are numerous.

According to the concepts described in [RFC3444], IEEE 1588-2008 itself provides an information model in its normative specifications for the data sets (in IEEE 1588-2008 clause 8). Some standardization organizations including the IETF have specified data models in MIBs (Management Information Bases) for IEEE 1588-2008 data sets (e.g. [RFC8173], [IEEE8021AS]). These MIBs are typically focused on retrieval of state data using the Simple Network Management Protocol (SNMP), furthermore, configuration of PTP data sets is not considered in [RFC8173].

Some service providers and applications require that the management of the IEEE 1588-2008 synchronization network be flexible and more Internet-based (typically overlaid on their transport networks). Software Defined Network (SDN) is another driving factor, which demands an improved configuration capability of synchronization networks.

YANG [RFC7950] is a data modeling language used to model configuration and state data manipulated by network management protocols like the Network Configuration Protocol (NETCONF) [RFC6241]. A small set of built-in data types are defined in [RFC7950], and a collection of common data types are further defined in [RFC6991]. Advantages of YANG include Internet based configuration capability, validation, rollback and so on. All of these characteristics make it attractive to become another candidate modeling language for IEEE 1588-2008.

This document defines a YANG data model for the configuration of IEEE 1588-2008 devices and clocks, and retrieval of the state data of IEEE 1588-2008 clocks. The data model is based on the PTP data sets as specified in [IEEE1588]. The technology specific IEEE 1588-2008 information, e.g., those specifically implemented by a bridge, a router or a telecom profile, is out of scope of this document.

The YANG module in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

When used in practice, network products in support of synchronization typically conform to one or more IEEE 1588-2008 profiles. Each profile specifies how IEEE 1588-2008 is used in a given industry (e.g. telecom, automotive) and application. A profile can require features that are optional in IEEE 1588-2008, and it can specify new features that use IEEE 1588-2008 as a foundation.

It is expected that the IEEE 1588-2008 YANG module be used as follows:

- o The IEEE 1588-2008 YANG module can be used as-is for products that conform to one of the default profiles specified in IEEE 1588-2008.
- o When the IEEE 1588 standard is revised (e.g. the IEEE 1588 revision in progress at the time of writing this document), it will add some new optional features to its data sets. The YANG module of this document can be revised and extended to support these new features. Moreover, the YANG "revision" MUST be used to indicate changes to the YANG module under such a circumstance.
- o A profile standard based on IEEE 1588-2008 may create a dedicated YANG module for its profile. The profile's YANG module SHOULD use YANG "import" to import the IEEE 1588-2008 YANG module as its foundation. Then the profile's YANG module SHOULD use YANG "augment" to add any profile-specific enhancements.
- o A product that conforms to a profile standard may also create its own YANG module. The product's YANG module SHOULD "import" the profile's module, and then use YANG "augment" to add any product-specific enhancements.

1.1. Conventions used in this document

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

1.2. Terminology

Most terminologies used in this document are extracted from [IEEE1588].

BC	Boundary Clock, see Section 3.1.3 of [IEEE1588]
DS	Data Set
E2E	End-to-End
EUI	Extended Unique Identifier
GPS	Global Positioning System

IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OC	Ordinary Clock, see Section 3.1.22 of [IEEE1588]
P2P	Peer-to-Peer
PTP	Precision Time Protocol
TAI	International Atomic Time
TC	Transparent Clock, see Section 3.1.46 of [IEEE1588]
UTC	Coordinated Universal Time
PTP data set	Structured attributes of clocks (an OC, BC or TC) used for PTP protocol decisions and for providing values for PTP message fields, see Section 8 of [IEEE1588].
PTP instance	A PTP implementation in the device (i.e., an OC or BC) represented by a specific PTP data set.

2. IEEE 1588-2008 YANG Model hierarchy

This section describes the hierarchy of an IEEE 1588-2008 YANG module. Query and configuration of device wide or port specific configuration information and clock data set are described for this version.

Query and configuration of clock information include:

(Note: The attribute names are consistent with IEEE 1588-2008, but changed to the YANG style, i.e., using all lower-case, with dashes between words.)

- Clock data set attributes in a clock node, including: current-ds, parent-ds, default-ds, time-properties-ds, and transparent-clock-default-ds.

- Port-specific data set attributes, including: port-ds and transparent-clock-port-ds.

The readers are assumed to be familiar with IEEE 1588-2008. As all PTP terminologies and PTP data set attributes are described in details in IEEE 1588-2008 [IEEE1588], this document only outlines each of them in the YANG module.

A simplified YANG tree diagram [RFC8340] representing the data model is typically used by YANG modules. This document uses the same tree diagram syntax as described in [RFC8340].

```

module: ietf-ptp
  +--rw ptp
    +--rw instance-list* [instance-number]
      +--rw instance-number      uint32
      +--rw default-ds
        +--rw two-step-flag?     boolean
        +--ro clock-identity?    clock-identity-type
        +--rw number-ports?      uint16
        +--rw clock-quality
          +--rw clock-class?      uint8
          +--rw clock-accuracy?   uint8
          +--rw offset-scaled-log-variance? uint16
        +--rw priority1?         uint8
        +--rw priority2?         uint8
        +--rw domain-number?     uint8
        +--rw slave-only?        boolean
      +--rw current-ds
        +--rw steps-removed?      uint16
        +--rw offset-from-master? time-interval-type
        +--rw mean-path-delay?    time-interval-type
      +--rw parent-ds
        +--rw parent-port-identity
          +--rw clock-identity?   clock-identity-type
          +--rw port-number?      uint16
        +--rw parent-stats?       boolean
        +--rw observed-parent-offset-scaled-log-variance? uint16
        +--rw observed-parent-clock-phase-change-rate?   int32
        +--rw grandmaster-identity? clock-identity-type
        +--rw grandmaster-clock-quality
          +--rw clock-class?      uint8
          +--rw clock-accuracy?   uint8
          +--rw offset-scaled-log-variance? uint16
        +--rw grandmaster-priority1? uint8
  
```

```

|   +--rw grandmaster-priority2?          uint8
+--rw time-properties-ds
|   +--rw current-utc-offset-valid?    boolean
|   +--rw current-utc-offset?          int16
|   +--rw leap59?                      boolean
|   +--rw leap61?                      boolean
|   +--rw time-traceable?              boolean
|   +--rw frequency-traceable?         boolean
|   +--rw ptp-timescale?               boolean
|   +--rw time-source?                 uint8
+--rw port-ds-list* [port-number]
|   +--rw port-number                  uint16
|   +--rw port-state?                  port-state-enum
|   +--rw underlying-interface?        if:interface-ref
|   +--rw log-min-delay-req-interval?  int8
|   +--rw peer-mean-path-delay?        time-interval-type
|   +--rw log-announce-interval?      int8
|   +--rw announce-receipt-timeout?    uint8
|   +--rw log-sync-interval?          int8
|   +--rw delay-mechanism?             delay-mechanism-enum
|   +--rw log-min-pdelay-req-interval? int8
|   +--rw version-number?              uint8
+--rw transparent-clock-default-ds
|   +--ro clock-identity?              clock-identity-type
|   +--rw number-ports?                uint16
|   +--rw delay-mechanism?             delay-mechanism-enum
|   +--rw primary-domain?              uint8
+--rw transparent-clock-port-ds-list* [port-number]
|   +--rw port-number                  uint16
|   +--rw log-min-pdelay-req-interval? int8
|   +--rw faulty-flag?                 boolean
|   +--rw peer-mean-path-delay?        time-interval-type

```

2.1. Interpretations from IEEE 1588 Working Group

The preceding model and the associated YANG module have some subtle differences from the data set specifications of IEEE Std 1588-2008. These differences are based on interpretation from the IEEE 1588 Working Group, and are intended to provide compatibility with future revisions of the IEEE 1588 standard.

In IEEE Std 1588-2008, a physical product can implement multiple PTP clocks (i.e., ordinary, boundary, or transparent clock). As specified in 1588-2008 subclause 7.1, each of the multiple clocks operates in an independent domain. However, the organization of multiple PTP domains was not clear in the data sets of IEEE Std 1588-2008. This document introduces the concept of PTP instance as described in the new revision of IEEE 1588. The instance concept is used exclusively to allow for optional support of multiple domains. The instance number has no usage within PTP messages.

Based on statements in IEEE 1588-2008 subclauses 8.3.1 and 10.1, most transparent clock products have interpreted the transparent clock data sets to reside as a singleton at the root level of the managed product, and this YANG model reflects that location.

2.2. Configuration and state

The information model of IEEE Std 1588-2008 classifies each member in PTP data sets as one of the following:

- Configurable: Writable by management.
- Dynamic: Read-only to management, and the value is changed by 1588 protocol operation.
- Static: Read-only to management, and the value typically does not change.

For details on the classification of each PTP data set member, refer to the IEEE Std 1588-2008 specification for that member.

Under certain circumstances, the classification of an IEEE 1588 data set member may change for a YANG implementation, for example, a configurable member needs to be changed to read-only. In such a case, an implementation SHOULD choose to return a warning upon writing to a read-only member, or use the deviation mechanism to develop a new deviation model as described in Section 7.20.3 of [RFC7950].

3. IEEE 1588-2008 YANG Module

This module imports typedef "interface-ref" from [RFC8343]. Most attributes are based on the information model defined in [IEEE1588], but their names are adapted to the YANG style of naming.

```
<CODE BEGINS> file "ietf-ptp@2018-09-10.yang"
//Note to RFC Editor: update the date to date of publication
module ietf-ptp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ptp";
  prefix "ptp";

  import ietf-interfaces {
    prefix if;
    reference
      "RFC8343: A YANG Data Model for Interface Management";
  }

  organization "IETF TICTOC Working Group";
  contact
    "WG Web:  http://tools.ietf.org/wg/tictoc/
    WG List:  <mailto:tictoc@ietf.org>
    Editor:   Yuanlong Jiang
              <mailto:jiangyuanlong@huawei.com>
    Editor:   Rodney Cummings
              <mailto:rodney.cummings@ni.com>";
  description
    "This YANG module defines a data model for the configuration
    of IEEE 1588-2008 clocks, and also for retrieval of the state
    data of IEEE 1588-2008 clocks.";

  revision "2018-09-10" {
    //Note to RFC Editor: update the date to date of publication
    description "Initial version";
    reference "RFC XXXX: YANG Data Model for IEEE 1588-2008";
    //Note to RFC Editor: update RFC XXXX to the actual RFC number
  }

  typedef delay-mechanism-enumeration {
    type enumeration {
      enum e2e {
        value 1;
        description
          "The port uses the delay request-response mechanism.";
      }
    }
  }
```



```
enum p2p {
    value 2;
    description
        "The port uses the peer delay mechanism.";
}
enum disabled {
    value 254;
    description
        "The port does not implement any delay mechanism.";
}
}
description
    "The propagation delay measuring option used by the
    port. Values for this enumeration are specified
    by the IEEE 1588 standard exclusively.";
reference
    "IEEE Std 1588-2008: 8.2.5.4.4";
}

typedef port-state-enumeration {
    type enumeration {
        enum initializing {
            value 1;
            description
                "The port is initializing its data sets, hardware, and
                communication facilities.";
        }
        enum faulty {
            value 2;
            description
                "The port is in the fault state.";
        }
        enum disabled {
            value 3;
            description
                "The port is disabled, and is not communicating PTP
                messages (other than possibly PTP management
                messages).";
        }
        enum listening {
            value 4;
            description
                "The port is listening for an Announce message.";
        }
        enum pre-master {
            value 5;
            description
```

```
        "The port is in the pre-master state.";
    }
    enum master {
        value 6;
        description
            "The port is behaving as a master port.";
    }
    enum passive {
        value 7;
        description
            "The port is in the passive state.";
    }
    enum uncalibrated {
        value 8;
        description
            "A master port has been selected, but the port is still
            in the uncalibrated state.";
    }
    enum slave {
        value 9;
        description
            "The port is synchronizing to the selected master port.";
    }
}

description
    "The current state of the protocol engine associated
    with the port.  Values for this enumeration are specified
    by the IEEE 1588 standard exclusively.";
reference
    "IEEE Std 1588-2008: 8.2.5.3.1, 9.2.5";
}

typedef time-interval-type {
    type int64;
    description
        "Derived data type for time interval, represented in units of
        nanoseconds and multiplied by 2^16";
    reference
        "IEEE Std 1588-2008: 5.3.2";
}

typedef clock-identity-type {
    type binary {
        length "8";
    }
    description
```

```
    "Derived data type to identify a clock";
  reference
    "IEEE Std 1588-2008: 5.3.4";
}

grouping clock-quality-grouping {
  description
    "Derived data type for quality of a clock, which contains
    clockClass, clockAccuracy and offsetScaledLogVariance.";
  reference
    "IEEE Std 1588-2008: 5.3.7";

  leaf clock-class {
    type uint8;
    default 248;
    description
      "The clockClass denotes the traceability of the time
      or frequency distributed by the clock.";
  }

  leaf clock-accuracy {
    type uint8;
    description
      "The clockAccuracy indicates the expected accuracy
      of the clock.";
  }

  leaf offset-scaled-log-variance {
    type uint16;
    description
      "The offsetScaledLogVariance provides an estimate of
      the variations of the clock from a linear timescale
      when it is not synchronized to another clock
      using the protocol.";
  }
}

container ptp {
  description
    "The PTP struct containing all attributes of PTP data set,
    other optional PTP attributes can be augmented as well.";

  list instance-list {

    key "instance-number";
```

```
description
  "List of one or more PTP data sets in the device (see IEEE
  Std 1588-2008 subclause 6.3).
  Each PTP data set represents a distinct instance of
  PTP implementation in the device (i.e., distinct
  Ordinary Clock or Boundary Clock).";

leaf instance-number {
  type uint32;
  description
    "The instance number of the current PTP instance.
    This instance number is used for management purposes
    only. This instance number does not represent the PTP
    domain number, and is not used in PTP messages.";
}

container default-ds {
  description
    "The default data set of the clock (see IEEE Std
    1588-2008 subclause 8.2.1). This data set represents
    the configuration/state required for operation
    of Precision Time Protocol (PTP) state machines.";

  leaf two-step-flag {
    type boolean;
    description
      "When set to true, the clock is a two-step clock;
      otherwise, the clock is a one-step clock.";
  }

  leaf clock-identity {
    type clock-identity-type;
    config false;
    description
      "The clockIdentity of the local clock";
  }

  leaf number-ports {
    type uint16;
    description
      "The number of PTP ports on the instance.";
  }

  container clock-quality {
    description
      "The clockQuality of the local clock.";
```

```
    uses clock-quality-grouping;
  }

  leaf priority1 {
    type uint8;
    description
      "The priority1 attribute of the local clock.";
  }

  leaf priority2{
    type uint8;
    description
      "The priority2 attribute of the local clock.";
  }

  leaf domain-number {
    type uint8;
    description
      "The domain number of the current syntonization
      domain.";
  }

  leaf slave-only {
    type boolean;
    description
      "When set to true, the clock is a slave-only clock.";
  }
}

container current-ds {
  description
    "The current data set of the clock (see IEEE Std
    1588-2008 subclause 8.2.2). This data set represents
    local states learned from the exchange of
    Precision Time Protocol (PTP) messages.";

  leaf steps-removed {
    type uint16;
    default 0;
    description
      "The number of communication paths traversed
      between the local clock and the grandmaster clock.";
  }

  leaf offset-from-master {
    type time-interval-type;
  }
}
```

```
        description
            "The current value of the time difference between
             a master and a slave clock as computed by the slave.";
    }

    leaf mean-path-delay {
        type time-interval-type;
        description
            "The current value of the mean propagation time between
             a master and a slave clock as computed by the slave.";
    }
}

container parent-ds {
    description
        "The parent data set of the clock (see IEEE Std 1588-2008
         subclause 8.2.3).";

    container parent-port-identity {
        description
            "The portIdentity of the port on the master, it
             contains two members: clockIdentity and portNumber.";
        reference
            "IEEE Std 1588-2008: 5.3.5";

        leaf clock-identity {
            type clock-identity-type;
            description
                "Identity of the clock";
        }

        leaf port-number {
            type uint16;
            description
                "Port number";
        }
    }

    leaf parent-stats {
        type boolean;
        default false;
        description
            "When set to true, the values of
             observedParentOffsetScaledLogVariance and
             observedParentClockPhaseChangeRate of parentDS
```

```
        have been measured and are valid.";
    }

    leaf observed-parent-offset-scaled-log-variance {
        type uint16;
        default 65535;
        description
            "An estimate of the parent clock's PTP variance
             as observed by the slave clock.";
    }

    leaf observed-parent-clock-phase-change-rate {
        type int32;
        description
            "An estimate of the parent clock's phase change rate
             as observed by the slave clock.";
    }

    leaf grandmaster-identity {
        type clock-identity-type;
        description
            "The clockIdentity attribute of the grandmaster clock.";
    }

    container grandmaster-clock-quality {
        description
            "The clockQuality of the grandmaster clock.";
        uses clock-quality-grouping;
    }

    leaf grandmaster-priority1 {
        type uint8;
        description
            "The priority1 attribute of the grandmaster clock.";
    }

    leaf grandmaster-priority2 {
        type uint8;
        description
            "The priority2 attribute of the grandmaster clock.";
    }

}

container time-properties-ds {
    description
        "The timeProperties data set of the clock (see
```

```
IEEE Std 1588-2008 subclause 8.2.4).";

leaf current-utc-offset-valid {
  type boolean;
  description
    "When set to true, the current UTC offset is valid.";
}
leaf current-utc-offset {
  when "../current-utc-offset-valid='true'";
  type int16;
  description
    "The offset between TAI and UTC when the epoch of the
    PTP system is the PTP epoch in units of seconds, i.e.,
    when ptp-timescale is TRUE; otherwise, the value has
    no meaning.";
}

leaf leap59 {
  type boolean;
  description
    "When set to true, the last minute of the current UTC
    day contains 59 seconds.";
}

leaf leap61 {
  type boolean;
  description
    "When set to true, the last minute of the current UTC
    day contains 61 seconds.";
}

leaf time-traceable {
  type boolean;
  description
    "When set to true, the timescale and the
    currentUtcOffset are traceable to a primary
    reference.";
}

leaf frequency-traceable {
  type boolean;
  description
    "When set to true, the frequency determining the
    timescale is traceable to a primary reference.";
}
```



```
leaf ptp-timescale {
  type boolean;
  description
    "When set to true, the clock timescale of the
    grandmaster clock is PTP; otherwise, the timescale is
    ARB
    (arbitrary).";
}

leaf time-source {
  type uint8;
  description
    "The source of time used by the grandmaster clock.";
}
}

list port-ds-list {
  key "port-number";
  description
    "List of port data sets of the clock (see IEEE Std
    1588-2008 subclause 8.2.5).";

  leaf port-number {
    type uint16;

    description
      "Port number.
      The data sets (i.e., information model) of IEEE Std
      1588-2008 specify a member portDS.portIdentity, which
      uses a typed struct with members clockIdentity and
      portNumber.

      In this YANG data model, portIdentity is not modeled
      in the port-ds-list, however, its members are provided
      as follows:
      portIdentity.portNumber is provided as this port-
      number leaf in port-ds-list; and
      portIdentity.clockIdentity is provided as the clock-
      identity leaf in default-ds of the instance
      (i.e., ../../default-ds/clock-identity).";
  }

  leaf port-state {
    type port-state-enum;
    default "initializing";
    description
      "Current state associated with the port.";
  }
}
```

```
}

leaf underlying-interface {
  type if:interface-ref;
  description
    "Reference to the configured underlying interface that
     is used by this PTP Port (see RFC 8343).";
}

leaf log-min-delay-req-interval {
  type int8;
  description
    "The base-two logarithm of the minDelayReqInterval
     (the minimum permitted mean time interval between
     successive Delay_Req messages).";
}

leaf peer-mean-path-delay {
  type time-interval-type;
  default 0;
  description
    "An estimate of the current one-way propagation delay
     on the link when the delayMechanism is P2P; otherwise,
     it is zero.";
}

leaf log-announce-interval {
  type int8;
  description
    "The base-two logarithm of the mean
     announceInterval (mean time interval between
     successive Announce messages).";
}

leaf announce-receipt-timeout {
  type uint8;
  description
    "The number of announceInterval that have to pass
     without receipt of an Announce message before the
     occurrence of the event ANNOUNCE_RECEIPT_TIMEOUT_
     EXPIRES.";
}

leaf log-sync-interval {
  type int8;
  description
    "The base-two logarithm of the mean SyncInterval
```

```
        for multicast messages. The rates for unicast
        transmissions are negotiated separately on a per port
        basis and are not constrained by this attribute.";
    }

    leaf delay-mechanism {
        type delay-mechanism-enumeration;
        description
            "The propagation delay measuring option used by the
            port in computing meanPathDelay.";
    }

    leaf log-min-pdelay-req-interval {
        type int8;
        description
            "The base-two logarithm of the
            minPdelayReqInterval (minimum permitted mean time
            interval between successive Pdelay_Req messages).";
    }

    leaf version-number {
        type uint8;
        description
            "The PTP version in use on the port.";
    }
}

container transparent-clock-default-ds {
    description
        "The members of the transparentClockDefault data set (see
        IEEE Std 1588-2008 subclause 8.3.2).";

    leaf clock-identity {
        type clock-identity-type;
        config false;
        description
            "The clockIdentity of the transparent clock.";
    }

    leaf number-ports {
        type uint16;
        description
            "The number of PTP ports on the transparent clock.";
    }
}
```

```
leaf delay-mechanism {
  type delay-mechanism-enumeration;
  description
    "The propagation delay measuring option
    used by the transparent clock.";
}

leaf primary-domain {
  type uint8;
  default 0;
  description
    "The domainNumber of the primary syntonization domain (see
    IEEE Std 1588-2008 subclause 10.1).";
}
}

list transparent-clock-port-ds-list {
  key "port-number";
  description
    "List of transparentClockPort data sets of the transparent
    clock (see IEEE Std 1588-2008 subclause 8.3.3).";

  leaf port-number {
    type uint16;
    description
      "Port number.
      The data sets (i.e., information model) of IEEE Std
      1588-2008 specify a member
      transparentClockPortDS.portIdentity, which uses a typed
      struct with members clockIdentity and portNumber.

      In this YANG data model, portIdentity is not modeled in
      the transparent-clock-port-ds-list, however, its
      members are provided as follows:
      portIdentity.portNumber is provided as this leaf member
      in transparent-clock-port-ds-list; and
      portIdentity.clockIdentity is provided as the clock-
      identity leaf in transparent-clock-default-ds
      (i.e., ../../transparent-clock-default-ds/clock-
      identity).";

  }

  leaf log-min-pdelay-req-interval {
    type int8;
```

```
    description
      "The logarithm to the base 2 of the
       minPdelayReqInterval (minimum permitted mean time
       interval between successive Pdelay_Req messages).";
  }

  leaf faulty-flag {
    type boolean;
    default false;
    description
      "When set to true, the port is faulty.";
  }

  leaf peer-mean-path-delay {
    type time-interval-type;
    default 0;
    description
      "An estimate of the current one-way propagation delay
       on the link when the delayMechanism is P2P; otherwise,
       it is zero.";
  }
}
}
```

<CODE ENDS>

4. 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]. Furthermore, general security considerations of time protocols are discussed in [RFC7384].

The NETCONF access control model [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 are writable, and the involved subtrees that are sensitive include:

/ptp/instance-list specifies an instance (i.e., PTP data sets) for an OC or BC.

/ptp/transparent-clock-default-ds specifies a default data set for a TC.

/ptp/transparent-clock-port-ds-list specifies a list of port data sets for a TC.

Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. Specifically, an inappropriate configuration of them may adversely impact a PTP synchronization network. For example, loss of synchronization on a clock, accuracy degradation on a set of clocks, or even break down of a whole synchronization network.

5. IANA Considerations

This document registers the following URI in the "IETF XML registry" [RFC3688]:

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

Registrant Contact: The IESG

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

This document registers the following YANG module in the "YANG Module Names" registry [RFC6020]:

Name: ietf-ptp

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

Prefix: ptp

Reference: RFC XXXX

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) ", RFC 6020, October 2010

- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and Bierman, A., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011
- [RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013
- [RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017
- [RFC8341] Bierman, A. and Bjorklund, M., "Network Configuration Protocol (NETCONF) Access Control Model", RFC 8341, March 2018
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, March 2018
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, March 2018
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018
- [IEEE1588] IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, July 2008

6.2. Informative References

- [IEEE8021AS] IEEE, "Timing and Synchronizations for Time-Sensitive Applications in Bridged Local Area Networks", IEEE 802.1AS-2001, 2011
- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003

- [RFC4663] Harrington, D., "Transferring MIB Work from IETF Bridge MIB WG to IEEE 802.1 WG", RFC 4663, September 2006
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, October 2014
- [RFC8340] Bjorklund, M., and Berger, L., "YANG Tree Diagrams", RFC 8340, March 2018
- [RFC8173] Shankarkumar, V., Montini, L., Frost, T., and Dowd, G., "Precision Time Protocol Version 2 (PTPv2) Management Information Base", RFC 8173, June 2017

7. Acknowledgments

The authors would like to thank Tom Petch, Radek Krejci, Mahesh Jethanandani, Tal Mizrahi, Opher Ronen, Liang Geng, Alex Campbell, Joe Gwinn, John Fletcher, William Zhao and Dave Thaler for their valuable reviews and suggestions, thank Benoit Claise and Radek Krejci for their validation of the YANG module, and thank Jingfei Lv and Zitao Wang for their discussions on IEEE 1588 and YANG respectively.

Appendix A Transferring YANG Work to IEEE 1588 WG

This Appendix is informational.

This appendix describes a future plan to transition responsibility for IEEE 1588 YANG modules from the IETF TICTOC Working Group (WG) to the IEEE 1588 WG, which develops the time synchronization technology that the YANG modules are designed to manage.

This appendix is forward-looking with regard to future standardization roadmaps in IETF and IEEE. Since those roadmaps cannot be predicted with significant accuracy, this appendix is informational, and it does not specify imperatives or normative specifications of any kind.

The IEEE 1588-2008 YANG module of this standard represents a cooperation between IETF (for YANG) and IEEE (for 1588). For the initial standardization of IEEE-1588 YANG modules, the information model is relatively clear (i.e., IEEE 1588 data sets), but expertise in YANG is required, making IETF an appropriate location for the standards. The TICTOC WG has expertise with IEEE 1588, making it the appropriate location within IETF.

The IEEE 1588 WG anticipates future changes to its standard on an ongoing basis. As IEEE 1588 WG members gain practical expertise with YANG, the IEEE 1588 WG will become more appropriate for standardization of its YANG modules. As the IEEE 1588 standard is revised and/or amended, IEEE 1588 members can more effectively synchronize the revision of this YANG module with future versions of the IEEE 1588 standard.

This appendix is meant to establish some clear expectations between IETF and IEEE about the future transfer of IEEE 1588 YANG modules to the IEEE 1588 WG. The goal is to assist in making the future transfer as smooth as possible. As the transfer takes place, some case-by-case situations are likely to arise, which can be handled by discussion on the IETF TICTOC WG mailing lists and/or appropriate liaisons.

This appendix obtained insight from [RFC4663], an informational memo that described a similar transfer of MIB work from the IETF Bridge MIB WG to the IEEE 802.1 WG.

A.1. Assumptions for the Transfer

For the purposes of discussion in this appendix, assume that the IESG has approved the publication of an RFC containing a YANG module for a published IEEE 1588 standard. As of this writing, this is IEEE Std 1588-2008, but it is possible that YANG modules for subsequent 1588 revisions could be published from the IETF TICTOC WG. For discussion in this appendix, we use the phrase "last IETF 1588 YANG" to refer to the most recently published 1588 YANG module from the IETF TICTOC WG.

The IEEE-SA Standards Board New Standards Committee (NesCom) handles new Project Authorization Requests (PARs) (see <http://standards.ieee.org/board/nes/>). PARs are roughly the equivalent of IETF Working Group Charters and include information concerning the scope, purpose, and justification for standardization projects.

Assume that IEEE 1588 has an approved PAR that explicitly specifies development of a YANG module. The transfer of YANG work will occur in the context of this IEEE 1588 PAR. For discussion in this appendix, we use the phrase "first IEEE 1588 YANG" to refer to the first IEEE 1588 standard for YANG.

Assume that as part of the transfer of YANG work, the IETF TICTOC WG agrees to cease all work on standard YANG modules for IEEE 1588.

Assume that the IEEE 1588 WG has participated in the development of the last IETF 1588 YANG module, such that the first IEEE 1588 YANG module will effectively be a revision of it. In other words, the transfer of YANG work will be relatively clean.

The actual conditions for the future transfer can be such that the preceding assumptions do not hold. Exceptions to the assumptions will need to be addressed on a case-by-case basis at the time of the transfer. This appendix describes topics that can be addressed based on the preceding assumptions.

A.2. Intellectual Property Considerations

During review of the legal issues associated with transferring Bridge MIB WG documents to the IEEE 802.1 WG (Section 3.1 and Section 9 of [RFC4663]), it was concluded that the IETF does not have sufficient legal authority to make the transfer to IEEE without the consent of the document authors.

If the last IETF 1588 YANG is published as a RFC, the work is required to be transferred from the IETF to the IEEE, so that IEEE 1588 WG can begin working on the first IEEE 1588 YANG.

When work on the first IEEE YANG module begins in the IEEE 1588 WG, that work derives from the last IETF YANG module of this RFC, requiring a transfer of that work from the IETF to the IEEE. In order to avoid having the transfer of that work be dependent on the availability of this RFC's authors at the time of its publication, the IEEE Standards Association department of Risk Management and Licensing provided the appropriate forms and mechanisms for this document's authors to assign a non-exclusive license for IEEE to create derivative works from this document. Those IEEE forms and mechanisms will be updated as needed for any future IETF YANG modules for IEEE 1588 (The signed forms are held by the IEEE Standards Association department of Risk Management and Licensing.). This will help to make the future transfer of work from IETF to IEEE occur as smoothly as possible.

As stated in the initial "Status of this Memo", the YANG module in this document conforms to the provisions of BCP 78. The IETF will retain all the rights granted at the time of publication in the published RFCs.

A.3. Namespace and Module Name

As specified in Section 5 "IANA Considerations", the YANG module in this document uses IETF as the root of its URN namespace and YANG module name.

Use of IETF as the root of these names implies that the YANG module is standardized in a Working Group of IETF, using the IETF processes. If the IEEE 1588 Working Group were to continue using these names rooted in IETF, the IEEE 1588 YANG standardization would need to continue in the IETF. The goal of transferring the YANG work is to avoid this sort of dependency between standards organizations.

IEEE 802 has an active PAR (IEEE P802d) for creating a URN namespace for IEEE use (see <http://standards.ieee.org/develop/project/802d.html>). It is likely that this IEEE 802 PAR will be approved and published prior to the transfer of YANG work to the IEEE 1588 WG. If so, the IEEE 1588 WG can use the IEEE URN namespace for the first IEEE 1588 YANG module, such as:

```
urn:ieee:Std:1588:yang:ieee1588-ptp
```

where "ieee1588-ntp" is the registered YANG module name in the IEEE.

Under the assumptions of section A.1, the first IEEE 1588 YANG module's prefix will be the same as the last IETF 1588 YANG module's prefix (i.e. "ntp"). Consequently, other YANG modules can preserve the same import prefix "ntp" to access PTP nodes during the migration from the last IETF 1588 YANG module to the first IEEE 1588 YANG module.

The result of these name changes are that for complete compatibility, a server (i.e., IEEE 1588 node) can choose to implement a YANG module for the last IETF 1588 YANG module (with IETF root) as well as the first IEEE 1588 YANG module (with IEEE root). Since the content of the YANG module transferred are the same, the server implementation is effectively common for both.

From a client's perspective, a client of the last IETF 1588 YANG module (or earlier) looks for the IETF-rooted module name; and a client of the first IEEE 1588 YANG module (or later) looks for the IEEE-rooted module name.

A.4. IEEE 1588 YANG Modules in ASCII Format

Although IEEE 1588 can certainly decide to publish YANG modules only in the PDF format that they use for their standard documents, without publishing an ASCII version, most network management systems cannot import the YANG module directly from the PDF. Thus, not publishing an ASCII version of the YANG module would negatively impact implementers and deployers of YANG modules and would make potential IETF reviews of YANG modules more difficult.

This appendix recommends that the IEEE 1588 WG consider future plans for:

- o Public availability of the ASCII YANG modules during project development. These ASCII files allow IETF participants to access these documents for pre-standard review purposes.
- o Public availability of the YANG portion of published IEEE 1588 standards, provided as an ASCII file for each YANG module. These ASCII files are intended for use of the published IEEE 1588 standard.

As an example of public availability during project development, IEEE 802 uses the same repository that IETF uses for YANG module development (see <https://github.com/YangModels/yang>). IEEE branches are provided for experimental work (i.e. pre-PAR) as well as

standard work (post-PAR drafts). IEEE-SA has approved use of this repository for project development, but not for published standards.

As an example of public availability of YANG modules for published standards, IEEE 802.1 provides a public list of ASCII files for MIB (see <http://www.ieee802.org/1/files/public/MIBs/> and <http://www.ieee802.org/1/pages/MIBS.html>), and analogous lists are planned for IEEE 802.1 YANG files.

Authors' Addresses

Yuanlong Jiang (Editor)
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China
Email: jiangyuanlong@huawei.com

Xian Liu
Independent
Shenzhen 518129, China
lene.liuxian@foxmail.com

Jinchun Xu
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China
xujinchun@huawei.com

Rodney Cummings (Editor)
National Instruments
11500 N. Mopac Expwy
Bldg. C
Austin, TX 78759-3504
Email: Rodney.Cummings@ni.com

TICTOC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 October 2022

D.A. Arnold
Meinberg-USA
H.G. Gerstung
Meinberg
6 April 2022

Enterprise Profile for the Precision Time Protocol With Mixed Multicast
and Unicast Messages
draft-ietf-tictoc-ptp-enterprise-profile-22

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 8 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

1. Introduction

The Precision Time Protocol ("PTP"), first 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 1588-2019 [IEEE1588] needs to know the identity of the time sources in the network (the Master Clocks). This document describes clock roles and port states using the terms master and slave in order to correspond to the terms used in IEEE 1588, on which this document is based. There is an active project in the IEEE to select alternative terms. When this project is completed, then master and slave will be replaced with the new alternative terms in an update to

this document.

The "Best Master Clock Algorithm" (IEEE 1588-2019 [IEEE1588] Subclause 9.3), a mechanism that all participating PTP nodes must follow, set up strict rules for all members of a PTP domain to determine which node 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 third edition of the standard (IEEE 1588-2019) defines PTPv2.1 and includes the possibility to use unicast communication between the PTP nodes in order to overcome the limitation of using multicast messages for the bi-directional information exchange between PTP nodes. The unicast approach avoided that, in PTP domains with a lot of nodes, devices had to throw away more than 99% of the received multicast messages because they carried information for some other node. PTPv2.1 also includes PTP profiles (IEEE 1588-2019 [IEEE1588] subclause 20.3). This construct allows organizations to specify selections of attribute values and optional features, simplifying the configuration of PTP nodes for a specific application. Instead of having to go through all possible parameters and configuration options and individually set them up, selecting a 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 1588-2019 [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 1588-2019 [IEEE1588] subclause 15.2) but allows a PTP profile specify an alternative management mechanism, for example NETCONF.

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 1588-2019 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.
- * PTPv2.1: Refers specifically to the third version of PTP defined by IEEE 1588-2019.
- * 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 configured 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:0:0:0:0:0:181 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 1588-2019 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 [RFC6241].

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

19. References

19.1. Normative References

- [IEEE1588] Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2019, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems.", November 2019, <<https://www.ieee.org>>.
- [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", February 2012, <<https://www.itu.int>>.
- [ISPCS] Arnold, D.A., "Plugfest Report", October 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
United States of America
Email: doug.arnold@meinberg-usa.com

Heiko Gerstung
Meinberg
Lange Wand 9
31812 Bad Pyrmont
Germany
Email: heiko.gerstung@meinberg.de

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: October 27, 2019

M. Lichvar
Red Hat
April 25, 2019

NTP Correction Field
draft-mlichvar-ntp-correction-field-04

Abstract

This document specifies an extension field for the Network Time Protocol (NTP) which improves resolution of specific fields in the NTP header and allows network devices such as switches and routers to modify NTP packets with corrections to improve accuracy of the synchronization in the network.

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 27, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

1. Introduction

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 [RFC5905].

If all network devices on the paths between NTP clients and servers implemented NTP and supported an operation as a server and client, the impact of the delays could be avoided by configuring NTP to make measurements only between devices and hosts that are directly connected to one another. In the Precision Time Protocol (PTP) [IEEE1588], which is a different protocol for synchronization of clocks in networks, such devices are called Boundary Clocks (BC).

A different approach supported by PTP to improve the accuracy uses Transparent Clocks (TC). Instead of fully implementing PTP in order to support an operation as a BC, the devices only modify a correction field in forwarded PTP packets with the time that the packets had to wait for transmission. The final value of the correction is included in the calculation of the delay and offset, which may significantly improve the accuracy and stability of the synchronization.

This document describes an NTP extension field which allows the devices to make a similar correction in forwarded NTP packets.

To better support a highly accurate synchronization, the extension field also improves resolution of the receive and transmit timestamps from the NTP header.

1.1. Requirements Language

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

2. Format of Correction Field

The Correction Field is an NTP extension field following RFC 7822 [RFC7822]. The format of the extension field is shown in Figure 1.

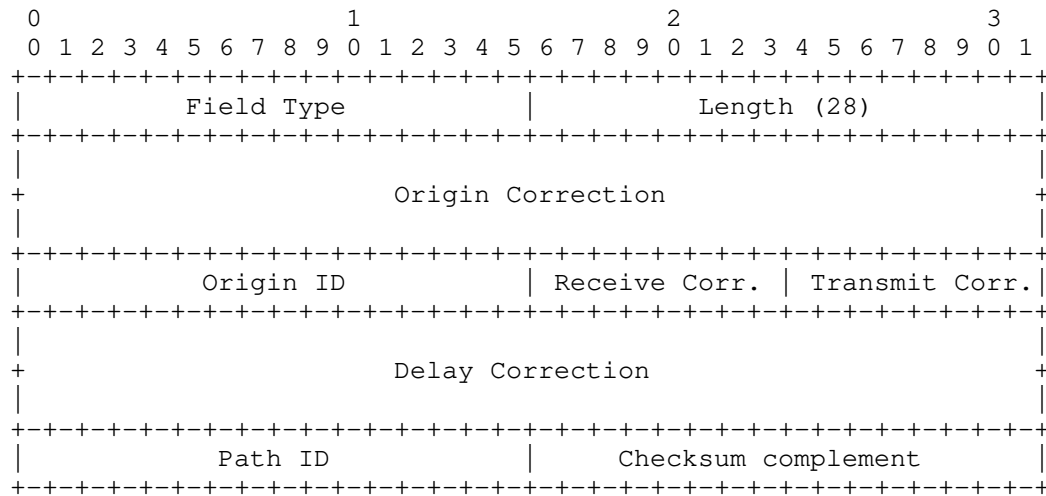


Figure 1: Format of Correction Field

The extension field has the following fields:

Field Type

The type which identifies the Correction extension field.
TBD

Length

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

Origin Correction

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

Origin ID

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

Receive Correction

An 8-bit extension of the receive timestamp in the NTP header increasing its resolution. The extended receive timestamp has 32 integer bits and 40 fractional bits.

Transmit Correction

An 8-bit extension of the transmit timestamp in the NTP header increasing its resolution. The extended transmit timestamp has 32 integer bits and 40 fractional bits.

Delay Correction

A signed fixed-point number of nanoseconds with 48 integer bits and 16 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].

3. Network devices

A network device which is forwarding a packet and supports the Correction Field MUST NOT modify the packet unless all of the following applies:

1. The packet is an IPv4 or IPv6 UDP packet.
2. The source port or destination port is 123.
3. The NTP version is 4.
4. The NTP mode is 1, 2, 3, 4, or 5.
5. The format of the packet is valid per RFC 7822.
6. The packet contains an extension field which has a type of TBD and length of 28 octets.

The device SHOULD add to the current value in the delay correction field the length of an interval between the reception and transmission of the packet. 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.

If the transmission starts before the reception ends, a negative value may need to be added to the delay correction. The end of the reception SHOULD be determined using the length field of the UDP header and the speed at which the packet is received.

If the device updates the delay correction, it SHOULD also add the identification numbers of the incoming and outgoing port to the path ID.

If the device 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.

4. NTP hosts

When an NTP client sends a request to a server and the association is configured to use the Correction Field, it SHOULD add the extension field to the packet. All fields of the extension field except type and length SHOULD be set to zero.

When the server receives a packet which includes the extension field, the response SHOULD also include the extension field.

If the server's clock has a better precision than resolution of the 64-bit NTP timestamp format, the server SHOULD save the additional bits in the receive and transmit correction fields and set the precision field to the corresponding number, which is smaller than -32. Otherwise, the receive and transmit correction fields SHOULD be zero.

The origin correction and origin ID fields SHOULD be set to the delay correction and path ID from the request. The other fields of the Correction Field SHOULD be zero.

When the client receives a response which contains the extension field, it SHOULD check the value of both the origin and delay correction fields. If a correction is larger than a specified maximum (e.g. 1 second), the extension field SHOULD be ignored.

The client MAY log a warning if the origin ID and path ID are not equal, which indicates the network path between the server and client is not symmetric.

If the client's clock has a better precision than resolution of the 64-bit NTP format and the precision field in the response contains a number smaller than -32, the client SHOULD extend the receive and transmit timestamp from the NTP header with the additional bits from the receive and transmit correction fields respectively.

When the client calculates the offset and delay using the formulas from RFC 5905, the origin correction is subtracted from the receive timestamp and the delay correction is added to the transmit timestamp. A conversion is necessary as the corrections are in different units than the timestamps (nanoseconds vs seconds).

An NTP peer follows the rules of both servers and clients. It processes Correction Fields in received packets as a client and sends Correction Fields as a server. A packet which has a zero origin timestamp (i.e. it is not a response to a request) SHOULD have a zero origin correction and zero origin ID in the Correction Field.

A broadcast server using the Correction Field SHOULD always set the origin correction and origin ID fields to zero.

5. Acknowledgements

The Correction Field extension is based on the PTP correction field specified in IEEE 1588-2008.

The author would like to thank Tal Mizrahi and Harlan Stenn for their useful comments.

6. IANA Considerations

IANA is requested to allocate an Extension Field Type for the Correction Field.

7. Security Considerations

NTP packets including the Correction Field cannot be authenticated by a legacy MAC, because the MAC has to cover all extension fields in the packet and devices which are supposed to modify the field are not able to update the MAC.

It is recommended to authenticate NTP packets using an authentication extension field, e.g. the NTS Authenticator and Encrypted Extensions [I-D.ietf-ntp-using-nts-for-ntp] extension field, and add the Correction Field to the packet after the authentication field.

A man-in-the-middle attacker can delay packets in the network in order to increase the measured delay and shift the measured offset by up to half of the extra delay. If the packets contain the Correction Field, the attacker can reduce the delay calculated by the client or peer and shift the offset even more. The maximum correction should be limited (e.g. to 1 second) to prevent the attacker from injecting a larger offset to the measurements.

8. References

8.1. 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>>.
- [RFC7822] Mizrahi, T. and D. Mayer, "Network Time Protocol Version 4 (NTPv4) Extension Fields", RFC 7822, DOI 10.17487/RFC7822, March 2016, <<https://www.rfc-editor.org/info/rfc7822>>.

8.2. Informative References

- [I-D.ietf-ntp-using-nts-for-ntp] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", draft-ietf-ntp-using-nts-for-ntp-18 (work in progress), April 2019.
- [IEEE1588] IEEE std. 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008.
- [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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 5, 2020

N. R.Schiff
D. Dolev
Hebrew University of Jerusalem
T. Mizrahi
Huawei Network.IO Innovation Lab
M. Schapira
Hebrew University of Jerusalem
September 2, 2019

A Secure Selection and Filtering Mechanism for the Network Time Protocol
Version 4
draft-schiff-ntp-chronos-03

Abstract

The Network Time Protocol version 4 (NTPv4) defines the peer process, the clock filter algorithm, the system process and the clock description algorithm. The clock filter algorithm and the system process, as defined in RFC 5905, are the mechanism according to which an NTP client chooses the NTP servers it synchronized with. This document specifies an alternative set of client mechanisms, named Chronos, that is backward compatible with NTPv4, and offers an improved level of security against time shifting attacks.

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 March 5, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Conventions Used in This Document	3
2.1. Terminology	3
2.2. Terms and Abbreviations	3
2.3. Notations	3
3. Extension for NTP Selection Process	4
3.1. Peer calibration Process	4
3.2. Chronos Selection Process	4
4. Chronos Pseudocode	5
5. Precision Vs. Security	5
6. Acknowledgements	6
7. IANA Considerations	6
8. Security Considerations	6
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Authors' Addresses	8

1. Introduction

According to RFC 5905 [RFC5905], the NTP servers used for updating the client's time are chosen by the clock filter algorithm and the system process. However, this method may be vulnerable to time shifting attacks, in which the attacker's goal is to shift the local time of an NTP client. Time shifting attacks on NTP are possible even if all NTP communications are encrypted and authenticated. This document introduces an improved system process with a secure algorithm called Chronos. Chronos is backwards compatible with NTPv4, as an NTP client that runs Chronos is interoperable with [RFC5905]-compatible NTPv4 servers.

Chronos achieves accurate synchronization even in the presence of powerful attackers who are in direct control of a large number of NTP servers. Chronos leverages ideas from distributed computing literature on clock synchronization in the presence of adversarial (Byzantine) behaviour.

A Chronos client iteratively "crowdsources" time queries across multiple NTP servers and applies a provably secure algorithm for eliminating "suspicious" responses and 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. The experimental results indicate that in order to implement a successful time-shifting attack on a Chronos client by over 100ms from the UTC, even a powerful man-in-the-middle attacker requires over 20 years of effort in expectation. The full paper is in [Chronos_paper].

Chronos differs from the current NTPv4 in two aspects. First, the Chronos client relies on a large number of NTP servers, from which only few are chosen at random in order to avoid overloading the servers. Second, the selection algorithm uses an approximate agreement technique to remove outliers, thus limiting the attacker's ability to contaminate the chosen time samples. These Chronos client mechanisms have provable security guarantees against man-in-the-middle attackers and attackers who are 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
w	An upper bound on the distance from the local time at any NTP server with an accurate clock ("truechimer" as in [RFC5905])
Cest	the client's estimate for the time that passed since its last synchronization to the server pool (sec)
ERR	$(2W * Cest) / 1000$
K	panic trigger
tc	the current time, as indicated by the client's local clock [sec]

Table 1: Chronos Notations

3. Extension for NTP Selection Process

A client that runs Chronos does not implement the functionality described in Sections 10 and 11 in [RFC5905]. Instead, the client implements the behavior described in this section and the next one.

3.1. Peer calibration Process

The peer calibration process gathers a server pool of hundreds of servers. Each NTP client conducts the peer process as in Section 9 in [RFC5905], on an hourly basis for 24 consecutive hours and generates the union of all received IP addresses. Importantly, this is executed in the background once in a long time (e.g., every few weeks/months).

3.2. Chronos Selection Process

The Chronos selection process samples the server pool and removes outliers (replaces the clock filter algorithm and the system process as in [RFC5905]). First, a subset on the order of tens of the servers in the server pool is selected at random. Then, out of the tens of collected samples, the third lowest-value samples and third highest value samples are discarded.

Given the remaining samples, Chronos checks two conditions:

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

- o The average value of the remaining samples is at a distance of at most $ERR+2w$ from the client's local clock.

(where w, ERR are described in Table 1).

In the event that both of these conditions are satisfied, the average of the remaining samples is the "final offset". Otherwise, a few tens of the servers from the pool are sampled again, in the exact same manner. This re-sampling process continues until the two conditions are finally 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".

In panic mode a Chronos client queries all the servers in the 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, which become the new "final offset".

As in [RFC5905], the final offset is passed to the clock discipline algorithm to steer the system clock to the correct time.

4. Chronos Pseudocode

The Chronos pseudocode Time Sampling Scheme is the following:

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

5. Precision Vs. Security

Chronos client changes the list of the sampled servers more frequently than NTPv4 [Chronos_paper], without using NTPv4 filters. This enables Chronos to be provably more secure than NTPv4 [RFC5905] but might adversely affect its precision and accuracy. Therefore we

add the following smoothing mechanism: Chronos returns the offset with minimal absolute value unless its distance from the average offset is larger than a predefined value. Another approach we considered was to use the same set of servers as in the previous sample, unless the difference between the current offset and the new offset is larger than a predefined value.

In our experiments we observed that with the smoothing mechanism, Chronos and NTP are similar in terms of precision and accuracy when there is no attack.

6. Acknowledgements

The authors would like to thank Miroslav Lichvar, Yaakov.J.Stein and Karen O'Donoghue for contributions to this document and helpful discussions and comments.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

As explained above, a Chronos client repeatedly gathers time samples from small subsets of a large pool of NTP servers. The following form of a man-in-the-middle (MitM) Byzantine attacker is considered: a MitM attacker is assumed to control a subset of the servers in the pool of available 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 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.

Analytical results (in [Chronos_paper]) indicate that in order to succeed in shifting time at a Chronos client by even a small time

shift (e.g., 100ms), even a powerful man-in-the-middle attacker requires many years of effort (e.g., over 20 years in expectation).

It should be noted that Chronos provides resilience to MitM attacks that cannot be achieved by cryptographic authentication protocols. However, adding an authentication and crypto-based security layer to the Chronos layer is important for achieving high security guarantees and detection of various spoofing and modification attacks.

Further details about the Chronos security considerations and guarantees are discussed in [Chronos_paper].

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, <http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_02A-2_Deutsch_paper.pdf>.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <<https://www.rfc-editor.org/info/rfc2629>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

[roughtime]

Patton, C., "Roughtime: Securing Time with Digital Signatures", 2018,
<<https://blog.cloudflare.com/roughtime/>>.

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
Obsoletes: 7822 (if approved)
Intended status: Standards Track
Expires: September 27, 2019

H. Stenn
D. Mills
Network Time Foundation
March 26, 2019

Network Time Protocol Version 4 (NTPv4) Extension Fields
draft-stenn-ntp-extension-fields-09

Abstract

Network Time Protocol version 4 (NTPv4) defines the optional usage of extension fields. An extension field, as defined in RFC 5905 [RFC5905] and RFC 5906 [RFC5906], resides after the end of the NTP header and supplies optional capabilities or information that cannot be conveyed in the basic NTP packet. This document updates RFC 5905 [RFC5905] by clarifying some points regarding NTP extension fields and their usage with legacy Message Authentication Codes (MACs), and removes wasteful requirements added by RFC 7822 [RFC7822].

This proposal deprecates RFC 7822 [RFC7822].

RFC EDITOR: PLEASE REMOVE THE FOLLOWING PARAGRAPH BEFORE PUBLISHING:

The source code and issues list for this draft can be found in <https://github.com/hstenn/ietf-ntp-extension-fields>

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 27, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Conventions Used in This Document	3
2.1. Requirements Language	4
2.2. Terms and Abbreviations	4
3. NTP MAC - RFC 5906 Update	4
3.1. RFC5906 Section 4. - Autokey Cryptography	4
3.2. RFC5906 Section 10. - Autokey Protocol Messages	4
3.3. RFC5906 Section 11.5. - Error Recovery	5
3.4. RFC5906 Section 13. - IANA Consideration	5
4. NTP Extension Fields - RFC 5905 Update	5
4.1. OLD: 'RFC5905 7.5 - NTP Extension Field Format'	5
4.2. NEW: 'RFC5905 Section 7.5 - NTP Extension Field Format'	6
4.3. NEW: 'RFC5905 Section 7.5.1 - Extension Fields and MACs'	8
4.4. OLD: 'RFC5905 Section 9.2. - Peer Process Operations'	10
4.5. NEW: 'RFC5905 Section 9.2. - Peer Process Operations'	10
5. Acknowledgements	10
6. IANA Considerations	10
7. Security Considerations	12
8. References	12
8.1. Normative References	12
8.2. Informative References	12
Authors' Addresses	12

1. Introduction

An NTP packet consists of a set of fixed fields that may be followed by optional fields. Two types of optional fields are defined: extension fields (EFs) as defined in Section 7.5 of RFC 5905 [RFC5905], and legacy Message Authentication Codes (legacy MACs).

If a legacy MAC is used, it resides at the end of the packet. This field can be either a 4-octet crypto-NAK or data that has traditionally been 16, 20 or 24 octets long.

Additional information about the content of a MAC is specified in RFC 5906 [RFC5906], but since that RFC is Informational an implementor that was not planning to provide Autokey would likely never read that document. The result of this would be interoperability problems, at least. To address this problem this proposal also copies and clarifies some of the content of RFC 5906, putting it into RFC 5905. Because there is a reasonable expectation that RFC 5906 will be deprecated, this document does not propose changes or updates to RFC 5906.

NTP extension fields are defined in RFC 5905 [RFC5905] as a generic mechanism that allows the addition of future extensions and features without modifying the NTP header format (Section 16 of RFC 5905 [RFC5905]).

With the knowledge and experience we have gained over time, it has become clear that simplifications, clarifications, and improvements can be made to the NTP specification around EFs and MACs.

This proposal adjusts and clarifies the requirements around EFs and MACs, allows EFs to be on 4-octet boundaries of any acceptable length, and provides methods to disambiguate packet parsing in the unexpected and unlikely case where an implementation would choose to send a packet that could be ambiguously parsed by the receiver.

This proposal deprecates RFC 7822 [RFC7822].

Implementations are still free to send EFs that are padded to longer lengths that otherwise follow the requirements below.

This document better specifies and clarifies extension fields as well as the requirements and parsing of a legacy MAC, with changes to address errors found after the publication of RFC 5905 [RFC5905] with respect to extension fields. Specifically, this document updates Section 7.5 of RFC 5905 [RFC5905], clarifying the relationship between extension fields and MACs, and expressly defines the behavior of a host that receives an unknown extension field.

2. Conventions Used in This Document

2.1. Requirements Language

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

2.2. Terms and Abbreviations

EF - Extension Field

MAC - Message Authentication Code

NTPv4 - Network Time Protocol, Version 4 RFC 5905 [RFC5905]

3. NTP MAC - RFC 5906 Update

This document copies and updates some information in RFC 5906 [RFC5906] and puts it in to RFC 5905, as follows:

3.1. RFC5906 Section 4. - Autokey Cryptography

This section describes some of the cryptography aspects of Autokey. The third paragraph describes the use of 128- and 160-bit message digests. The enumeration of 128- and 160-bit message digests is not meant to be limiting - other message digest lengths MAY be implemented. This paragraph also describes some of the expected semantic ranges of the key ID. This information belongs in RFC 5905. The key ID value is particularly significant because it provides additional detection and disambiguation protection when deciding if the next data portion is either a legacy MAC or an extension field. [This is additional evidence that although RFC 5906 is Informational, parts of its content are REQUIRED for proper behavior of RFC 5905.]

3.2. RFC5906 Section 10. - Autokey Protocol Messages

This section describes the extension field format, including initial flag bits, a Code field, and 8-bit Field Type, and the 16-bit Length. This proposal expands and clarifies this information and puts it into RFC 5905.

This section says "The reference implementation discards any packet with a field length of more than 1024 characters." but this is no longer true.

3.3. RFC5906 Section 11.5. - Error Recovery

This section describes the crypto-NAK, which should be described in RFC 5905. A crypto-NAK is used by RFC 5905 as well. [This is additional evidence that even though RFC 5906 was Informational, some of its content is REQUIRED for proper behavior for RFC 5905.]

3.4. RFC5906 Section 13. - IANA Consideration

This section lists the Autokey-related Extension Field Types, including Flag Bits, Codes, and Field Types, which should be described in RFC 5905, or perhaps in some other document. [This is additional evidence that even though RFC 5906 is Informational, some of its content is REQUIRED for proper behavior for RFC 5905.]

4. NTP Extension Fields - RFC 5905 Update

This document updates Section 7.5 of RFC 5905 [RFC5905] as follows:

4.1. OLD: 'RFC5905 7.5 - NTP Extension Field Format'

In NTPv4, one or more extension fields can be inserted after the header and before the MAC, which is always present when an extension field is present. Other than defining the field format, this document makes no use of the field contents. An extension field contains a request or response message in the format shown in Figure 14.

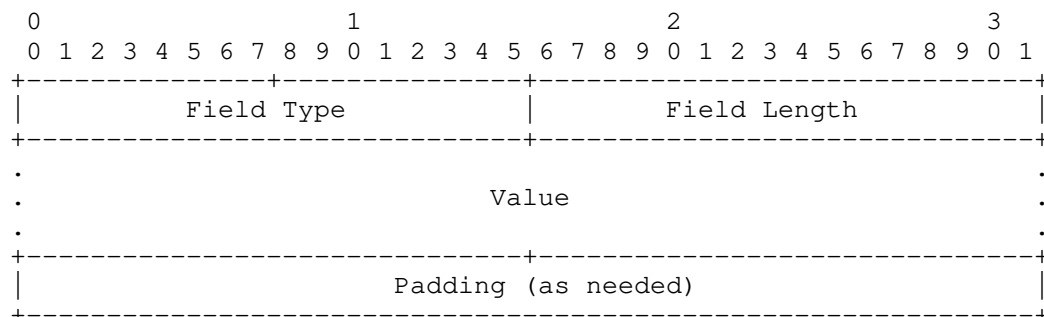


Figure 14: Extension Field Format

All extension fields are zero-padded to a word (four octets) boundary. The Field Type field is specific to the defined function and is not elaborated here. While the minimum field length containing required fields is four words (16 octets), a maximum field length remains to be established.

The Length field is a 16-bit unsigned integer that indicates the length of the entire extension field in octets, including the Padding field.

4.2. NEW: 'RFC5905 Section 7.5 - NTP Extension Field Format'

In NTPv4, one or more extension fields can be inserted after the header and before the possibly optional legacy MAC. A MAC SHOULD be present when an extension field is present. A MAC is always present in some form when NTP packets are authenticated. This MAC SHOULD be either a legacy MAC or a MAC-EF. It MAY be both. Other than defining the field format, this document makes no use of the field contents. An extension field contains a request or response message in the format shown in Figure 14.

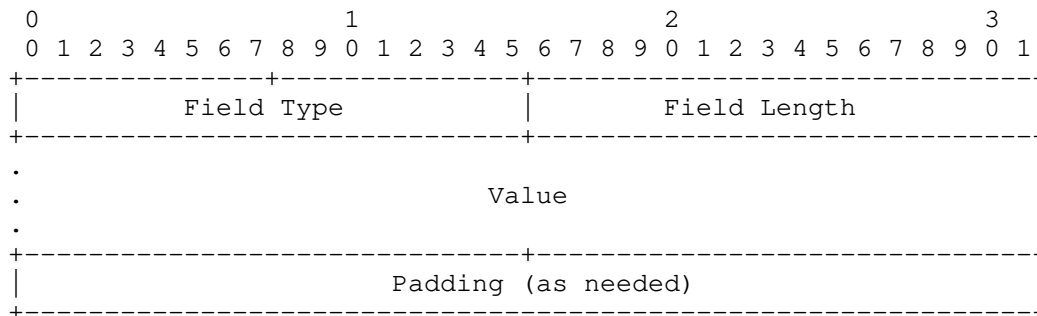


Figure 14: Extension Field Format

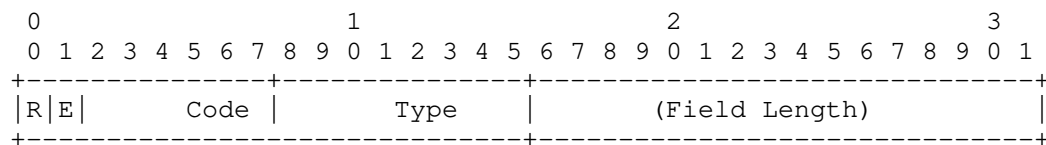
The four octets that comprise the Field Type and Field Length are called the Extension Field Header. Octets beyond the Extension Field Header are called the Extension Field Body, or the Extension Field Payload. The EF Body (EF Payload) MAY be null in some cases.

All extension fields are zero-padded to a word (four octet) boundary. The Field Type is specific to the defined functionality and detailed information about the Field Type is not elaborated here. The minimum size of an Extension Field is a 32-bit word (4 octets), and while the maximum extension field size MUST be 65532 octets or less, an NTP packet SHOULD NOT exceed the network MTU.

The Field Length is a 16-bit unsigned integer that indicates the length of the entire extension field in octets, including any Padding octets. The bottom two bits of the Field Length SHOULD be zero, and the size of the extension field SHOULD end on a 32-bit (4 octet) boundary. [RFC5905 Section 7.5 says "All extension fields are zero-padded to a word (four octets) boundary." but does not use 'MUST' language. Is it overkill to reiterate this requirement here? Should

we use SHOULD or MUST regarding the bottom two bits or the boundary of the EF? It is possible, down the road, that we might find some use for those bottom 2 bits, even if we require a 32-bit boundary on the last octet of an EF.]

The Field Type contains the following sub-elements:



Extension Field Header Format

Where the following Field Type flags are defined:

R: 0 for "Information/Query", 1 for a "Response"

E: 0 for "OK", 1 for an "Error". Unused, and will be deprecated.

[The 'R' flag is currently used by Autokey [RFC5906], and by the proposed I-DO [DRAFT-I-DO] extension field. This flag is used after the packet is accepted.]

[The 'E' flag was proposed for use by Autokey, after the packet was accepted. As it was never used and no other use-cases have been identified, we are recommending this flag be deprecated at some point in the future.]

[The EF Code subtype is currently used by RFC 5906, Autokey [RFC5906], by the proposed Extended Information [DRAFT-EXTENDED-INFORMATION], I-DO [DRAFT-I-DO], and is expected to be used by the NTS Extension Field, at least.]

The Autokey EF currently uses the most Code values - 10 of them, which equates to the least-significant 4 bits of the high-order octet. It is possible that additional flag bits will be allocated; in the past, the high-order 2 bits were reserved, and for a time two additional bits were proposed. Make no assumptions about the unused bits in this octet.

The EF Header and Body fields (the Flags, Code, Type, and Length, and any Value or Padding) are specific to the defined functionality and are not elaborated here; appropriate Field Type Flags, the EF Code, and EF Type values are defined in an IANA registry, and the Length, Value, and Padding values are defined by the document referred to by the registry. If a host receives an extension field with an unknown

Field Type, the host SHOULD ignore the extension field and MAY drop the packet altogether, depending on local policy.

The Length field is a 16-bit unsigned integer that indicates the length of the entire extension field in octets, including any Padding.

While the minimum field length of an EF that contains no value or padding fields is one word (four octets), and the minimum field length of an EF that contains required fields is two words (8 octets), the maximum field length MUST NOT be longer than 65532 octets due to the maximum size of the data represented by the Length field, and SHOULD be small enough that the size of the NTP packet received by the client does not exceed the smallest MTU between the sender and the recipient. The bottom two bits of the Field Length SHOULD be zero and the EF data SHOULD be aligned to a 32-bit (4 octet) boundary.

4.3. NEW: 'RFC5905 Section 7.5.1 - Extension Fields and MACs'

With the inclusion of additional Extension Fields, there is now a potential that a poorly-designed implementation would produce an ambiguous parsing in the presence of a legacy MAC. What follows are two possibly independent ways to prevent this situation from ever happening.

Note well that to-date, there are only two defined Extension Field Types: Autokey, defined by RFC 5906 [RFC5906], and the Experimental UDP Checksum Complement in the Network Time Protocol, defined by RFC 7821 [RFC7821].

In spite of its known serious problems, Autokey is still in use by some and is a legacy case that is easily supported. Old systems will still work. An old system will still be able to open a properly-configured Autokey association to a new system, a new system will still be able to open a properly-configured Autokey association with an old system, and two new systems will be able to open a properly-configured Autokey association.

The UDP Checksum Complement extension field forbids the use of a legacy MAC, so any packet that uses it CANNOT be using a legacy MAC. [We could list the detailed and specific reasons why traffic using this EF is immune to EF/legacy MAC problems, but I fear that would just be confusing to most people.]

The first and best way to prevent ambiguous parsing is to use the I-DO [DRAFT-I-DO] extension field.

By definition any NTP client or server that handles any other Extension Fields is "new code" and can completely prevent ambiguity by the initiating side sending a packet containing an I-DO [DRAFT-I-DO] extension field followed by an optional MAC-EF [DRAFT-MAC-LAST-EF] followed by an optional legacy MAC. The inclusion of any MAC would be dictated by the authentication requirements of the association.

Note that NTP traffic works perfectly well without using any other extension fields. Newer extension fields offer additional capabilities, but these capabilities are not required for operation. [Even in the case of NTS or SNT, we're talking about "new code" that can be expected to be aware of issues with new extension fields and legacy MACs.]

If the initiating side sends an I-DO [DRAFT-I-DO] packet and gets no response, it operates as if the other side cannot handle new extension fields and simply continues the association without sending any new extension fields. At any point in the future a packet can be sent with an I-DO extension field to see if the other side will respond.

An NTP implementation that receives a packet with an I-DO extension field may respond with a packet that may or may not contain an I-DO Response. If it does not respond, the other side SHOULD assume that the receiver does not understand new EFs. If it responds without sending an I-DO Response extension field, the sending side knows it should not send any new extension fields to this server. If the system that receives an I-DO extension field responds with an I-DO Response, it's telling the sender exactly what capabilities it is currently willing to exchange.

The second way to prevent ambiguous parsing is to use the LAST-EF [DRAFT-MAC-LAST-EF] extension field.

By definition, if I-DO is used and each side agrees to support LAST-EF then LAST-EF will prevent any ambiguity.

If, however, I-DO is not used then one side can simply send a packet with a LAST-EF. The LAST-EF extension field could be four-octet extension field, it could be a 28 octet extension field, or some other length that ends on a 32-bit boundary. If the other side responds appropriately then all is well. If the other side does not respond appropriately the sender should proceed without sending any new extension fields.

Parties interested in additional reasons for and approaches to understanding why there is no reason to be concerned about potential

ambiguities with new code that would use new extension fields and legacy MACs can look at the the drafts that preceded this document.

4.4. OLD: 'RFC5905 Section 9.2. - Peer Process Operations'

...

FXMIT. ... This message includes the normal NTP header data shown in Figure 8, but with a MAC consisting of four octets of zeros. ...

4.5. NEW: 'RFC5905 Section 9.2. - Peer Process Operations'

...

FXMIT. ... This message includes the normal NTP header data shown in Figure 8, but with a MAC consisting of four octets of zeros. This MAC can be a legacy MAC or a MAC-EF. If it's a MAC-EF, the crypto-NAK MUST be the only MAC in the MAC-EF payload. ...

5. Acknowledgements

The authors wish to acknowledge the contributions of Sam Weiler, Danny Mayer, and Tal Mizrahi.

6. IANA Considerations

This memo requests IANA to update the NTP Extension Field Types table in the NTP Parameters document as follows. The following is expected to be a functional superset of the existing information:

0										1											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5						
+-----+										+-----+											
R E		Code										Type									
+-----+										+-----+											

NTP Extension Field Type Format

Where the following Field Type flags are defined:

R: 0 for "Information/Query", 1 for a "Response"

E: 0 for "OK", 1 for an "Error". Unused, and will be deprecated.

Field Type	Meaning
0x0000	crypto-NAK (with Field Length of 0)
0x0000	RESERVED: Permanently Unassigned
0x0001	RESERVED: Unassigned
0x0002	Autokey: No-Operation Request
0x8002	Autokey: No-Operation Response
0x0102	Autokey: Association Message Request
0x8102	Autokey: Association Message Response
0x0202	Autokey: Certificate Message Request
0x8202	Autokey: Certificate Message Response
0x0302	Autokey: Cookie Message Request
0x8302	Autokey: Cookie Message Response
0x0402	Autokey: Autokey Message Request
0x8402	Autokey: Autokey Message Response
0x0502	Autokey: Leapseconds Value Message Request
0x8502	Autokey: Leapseconds Value Message Response
0x0602	Autokey: Sign Message Request
0x8602	Autokey: Sign Message Response
0x0702	Autokey: IFF Identity Message Request
0x8702	Autokey: IFF Identity Message Response
0x0802	Autokey: GQ Identity Message Request
0x8802	Autokey: GQ Identity Message Response
0x0902	Autokey: MV Identity Message Request
0x8902	Autokey: MV Identity Message Response
0x0003	* MAC
0x0104	* NTS Unique Identifier Request
0x8104	* NTS Unique Identifier Response
0x0204	* NTS Cookie
0x0304	* NTS Cookie Placeholder
0x0404	* NTS AEEF Request
0x8404	* NTS AEEF Response
0x0005	Checksum Complement
0x2005	Checksum Complement (deprecated flag 0x2000)
0x0006	* Suggest REFID
0x0007	* I-DO
0x0008	* LAST-EF
0x0009	* Extended Information
0x00FF	* thru
0xFDFF	* RESERVED for future I-DO Payloads
0xFEFF	* I-DO Payload: Leap Smear REFIDs
0xFFFF	* I-DO Payload: IPv6 REFID hash

* Tentative Reservation

Current Extension Fields

7. Security Considerations

The authors know of no adverse consequences of adopting this proposal.

8. References

8.1. 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>>.
- [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>>.

8.2. Informative References

- [DRAFT-EXTENDED-INFORMATION]
Stenn, H., "draft-stenn-ntp-extended-information", 2019.
- [DRAFT-I-DO]
Stenn, H., "draft-stenn-ntp-i-do", 2019.
- [DRAFT-MAC-LAST-EF]
Stenn, H., "draft-stenn-ntp-mac-last-ef", 2019.

Authors' Addresses

Harlan Stenn
Network Time Foundation
P.O. Box 918
Talent, OR 97540
US

Email: stenn@nwttime.org

David L. Mills
Network Time Foundation
P.O. Box 918
Talent, OR 97540
US

Email: mills@udel.edu