Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: June 12, 2021

M. Lichvar
Red Hat

Dec 9, 2020

# Network Time Protocol Version 5
# draft-mlichvar-ntp-ntpv5-01

## Abstract

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on June 12, 2021.

## Copyright Notice

Table of Contents



## 1. Introduction

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

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

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

2. NTPv5 drops support for the symmetric active, symmetric passive, broadcast, control, and private modes. Only the client and server modes are supported.

3. Timestamps are clearly separated from values used as cookies.

4. NTPv5 messages can be extended only with extension fields. The MAC field is wrapped in an extension field.

5. Extension fields can be of any length, even indivisible by 4, but are padded to a multiple of 4 octets. Extension fields specified for NTPv4 are compatible with NTPv5.

6. NTPv5 adds support for other timescales than UTC.

7. The NTP era number is exchanged in the protocol, which extends the unambiguous interval of the client from 136 years to about 35000 years.

8. NTPv5 adds a new measurement mode to provide clients with more accurate transmit timestamps.

9. NTPv5 works with sets of reference IDs to prevent synchronization loops over multiple hosts.

10. Resolution of the root delay and root dispersion fields is improved.

### 1.1. Requirements Language

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

## 2. Basic Concepts

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

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

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

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

Servers have randomly generated reference IDs to prevent synchronization loops.

## 3. Data Types

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

time16
: A 16-bit fixed-point type containing values in seconds. It has 1 signed integer bit (i.e. it is just the sign) and 15 fractional bits. The minimum value is -1.0, the maximum value is 32767/32768, and the resolution is about 30 microseconds.

time32
: A 32-bit fixed-point type containing values in seconds. It has 4 unsigned integer bits and 28 fractional bits. The maximum value is 16 seconds and the resolution is about 3.7 nanoseconds.

timestamp64
: A 64-bit fixed-point type containing timestamps. It has 32 signed integer bits and 32 fractional bits. It spans an interval of about 136 years and has a resolution of about 0.23 nanoseconds. It can be used in different timescales. In the UTC timescale it is the number of SI seconds since 1 Jan 1972 plus 2272060800, excluding leap seconds. Timestamps in the TAI timescale are the same except they include leap seconds and extra 10 seconds for the original difference between TAI and UTC in 1972, when leap seconds were introduced. One interval covered by the type is called an NTP era. The era starting at the epoch is era number 0, the following era is number 1, and so on.

## 4. Message Format

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|LI | VN  |Mode | Scale |Stratum|     Poll      |  Precision    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |      Era      |        Timescale Offset       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Root Delay                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Root Dispersion                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                        Server Cookie (64)                     +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                        Client Cookie (64)                     +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      Receive Timestamp (64)                   +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      Transmit Timestamp (64)                  +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                    Extension Field 1 (variable)               .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                    Extension Field N (variable)               .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Format of NTPv5 messages

Each NTPv5 message has a header containing the following fields:

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

Version Number (VN)
A 3-bit field containing the value 5.

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

Scale
A 4-bit identifier of the timescale. In requests it is the requested timescale. In responses it is the timescale of the receive and transmit timestamps. Defined values are:

0: UTC

1: TAI

2: UT1

3: Leap-smeared UTC

Stratum
A 4-bit field containing the stratum of host. Primary time servers have a stratum of 1, their clients have a stratum of 2, and so on. The value of 0 indicates an unknown or infinite stratum. In requests it is always 0.

Poll
An 8-bit signed integer containing the polling interval as a rounded log2 value in seconds. In requests it is the current polling interval. In responses it is the minimum allowed polling interval.

Precision
An 8-bit signed integer containing the precision of the timestamps included in the message as a rounded log2 value in seconds. In requests, which don't contain any timestamps, it is always 0.

Flags
An 8-bit integer that can contain the following flags:

0x1: Unknown leap

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

0x4: Interleaved mode
In requests it is a request for a response in the interleaved mode. In responses it indicates the response is in the interleaved mode.

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

Timescale Offset
A 16-bit value specific to the selected timescale, which is referenced to the receive timestamp. In requests it is always 0.

* In the UTC (0) and TAI (1) timescales it is the TAI-UTC offset as a signed integer, or 0x8000 if unknown.

* In the UT1 timescale (2) it is the UT1-UTC offset using the time16 type, or 0x8000 (-1.0) if unknown.

* In the leap-smeared UTC, it is the current offset between the leap smeared time and UTC using the time16 type, or 0x8000 (-1.0) if unknown.

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

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

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

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

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

Transmit Timestamp
A field using the timestamp64 type. In requests it is always 0. In responses it is the time when a response to the client was transmitted. The specific response depends on the selected mode (basic or interleaved). The timestamp corresponds to the beginning of the transmission.

The header has 48 octets, which is the minimum length of a valid NTPv5 message. A message can contain zero, one, or multiple extension fields. The maximum length is not specified, but the length is always divisible by 4.

## 5. Extension Fields

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                           Data (variable)                     .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Format of NTPv5 extension fields

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

Generally, if a request contains an extension field, the client is asking the server to include the same extension field in the response. Exceptions to this rule are allowed.

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

The rest of this section describes new extension fields specified for NTPv5. Clients are not required to use or support any of these

extension fields, but servers are required to support some extension fields.

### 5.1. Padding Extension Field

This field is used by servers to pad the response to the same length as the request if the response doesn't contain all requested extension fields, or some have a variable length. It can have any length.

This field MUST be supported on server.

### 5.2. MAC Extension Field

This field authenticates the NTPv5 message with a symmetric key. Implementations SHOULD use the MAC specified in RFC8573 [RFC8573]. The extension field MUST be the last extension field in the message unless an extension field is specifically allowed to be placed after a MAC or another authenticator field.

### 5.3. Reference IDs Extension Field

This field allows servers to prevent synchronization loops, i.e. synchronizing to one of its direct or indirect clients. It contains a set (bloom filter) of reference IDs.

TODO

This field MUST be supported on server.

### 5.4. Server Information Extension Field

This field provides clients with information about which NTP versions are supported by the server, as a minimum and maximum version. The extension field has a fixed length of 8 octets. In requests, all data fields of the extension are 0.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Min. Version  | Max. Version  |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Format of Server Information Extension Field

This field MUST be supported on server.

### 5.5. Correction Extension Field

This field allows switches and routers to make corrections in NTPv5 messages to allow clients to compensate for queueing and processing delays in the network.

TODO (reuse draft-mlichvar-ntp-correction-field?)

### 5.6. Reference Timestamp Extension Field

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                      Reference Timestamp (64)                 |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Format of Reference Timestamp Extension Field

### 5.7. Monotonic Timestamp Extension Field

This field contains an extra receive timestamp with a 32-bit epoch identifier from a clock which doesn't have adjusted phase and can be used for a frequency transfer, e.g. to stabilize synchronization in a long chain of servers. It has a constant length of 16 octets. In requests, the counter and timestamp are always 0.

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Epoch ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                  Monotonic Receive Timestamp (64)             |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Format of Monotonic Timestamp Extension Field

## 6. Client Operation

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

The polling interval can be adjusted for the network conditions and stability of the clock. When polling a public server on Internet, the client SHOULD use at least a polling interval of 64 seconds, increasing up to at least 1024 seconds.

The client can make measurements in the basic or interleaved mode. The interleaved mode allows the server to provide the client with a more accurate transmit timestamp which is available only after the server formed or sent the response.

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

On each poll, the client:

1. Generates a new random cookie.

2. Formats a request with necessary extension fields and the fields in the header all zero except:

   * Version is set to 5.

   * Mode is set to 3.

* Scale is set to the timescale in which the client wants to operate.

* Poll is set to the rounded log2 value of the current client's polling interval in seconds.

* Flags are set according to the requested mode. The interleaved mode flag requests a response in the interleaved mode.

* Server cookie is set only in the interleaved mode. If a valid response from the server was received previously, it is set to the server cookie from the previous response.

* Client cookie is set to the newly generated cookie.

3. Sends the request to the server to the UDP port 123 and captures a transmit timestamp.

4. Waits for a valid response from the server and captures a receive timestamp. A valid response has version 5, mode 4, client cookie equal to the cookie from the request, and passes authentication if enabled. The client MUST ignore all invalid responses and accept at most one valid response.

5. Checks whether the response is usable for synchronization of the clock. Such a response has a leap indicator not equal to 3, stratum between 0 and 16, root delay and dispersion both smaller than a specific value, e.g. 16 seconds, and timescale equal to the requested timescale. If the response is in a different timescale, the client can switch to the provided timescale, convert the timestamps if the offset between the timescales is provided or known, or drop the response.

6. Saves the server's receive and transmit timestamps. If the client internally counts seconds using a type wider than 32 bits, it SHOULD expand the timestamps with the provided NTP era.

7. Calculates the offset, delay, and dispersion:

   offset = ((T2 + T3) - (T4 + T1)) / 2

   delay = |(T4 - T1) - (T3 - T2)|

   dispersion = |T4 - T1| * DR

   where

T1 is the client's transmit timestamp

T2 is the server's receive timestamp

T3 is the server's transmit timestamp

T4 is the client's receive timestamp

DR is the client's estimated dispersion rate

## 7. Server Operation

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

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

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

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

- o Stratum SHOULD be one larger than stratum of the best server it uses for its own synchronization.

- o Root delay SHOULD be the best server's root delay in addition to the measured delay to the server.

- o Root dispersion SHOULD be the best server's root dispersion in addition to an estimate of the maximum drift of its own clock since the last update of the clock.

The server has a randomly generated reference ID and it MUST track reference IDs of its servers using the Reference IDs Extension Field.

For each received request, the server:

1. Captures a receive timestamp.

2. Checks the version in the request. If it is not equal to 5, it MUST either drop the request, or handle it according to the

specification corresponding to the protocol version. The server MAY respond with an NTPv5 message if and only if the request has version 5.

3. Drops the request if the format is not valid, mode is not 3, or authentication fails if the MAC Extension Field or another authenticator field is present.

4. Server forms a response with requested extension fields and sets the fields in the header as follows:

   * Leap Indicator, Stratum, Root delay, and Root dispersion, are set to the current server's values.

   * Version is set to 5.

   * Scale is set to the client's requested timescale if it is supported by the server. If not, the server SHOULD respond in any timescale it supports.

   * The flags are set as follows:

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

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

   * Era is set to the NTP era of the receive timestamp.

   * Timescale Offset is set to the timescale-specific offset, or 0x8000 if unknown.

   * Server Cookie is set when the interleaved mode is requested and it is supported by the server, even if the response cannot be in the requested mode yet due to the request having an invalid server cookie. The cookie identifies a more accurate transmit timestamp, which can be retrieved by the client later with another request.

   * Client Cookie is set to the Client Cookie from the request.

   * Receive Timestamp is set to the server's receive timestamp of the request.

* Transmit Timestamp is set to a value which depends on the measurement mode. In the basic mode it is the server's current time when the message if formed. In the interleaved mode it is the transmit timestamp of the previous response identified by the server cookie in the request, captured at some point after the message was formed.

5. Drops the response if it is longer than the request to prevent traffic amplification.

6. Sends the response.

7. Saves the transmit timestamp and server cookie, if the interleaved mode was requested and is supported by the server.

## 8. NTPv5 Negotiation in NTPv4

NTPv5 messages are not compatible with NTPv4, even if they do not contain any extension fields. Some widely used NTPv4 implementations are known to ignore the version and interpret all requests as NTPv4. Their responses to NTPv5 requests have a zero client cookie, which means they fail the client's validation and are ignored.

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

An NTP server which supports both NTPv4 and NTPv5 SHOULD check the reference timestamp in all NTPv4 client requests. If the reference timestamp contains the value 0x4E5450354E545035 ("NTP5NTP5" in ASCII), it SHOULD respond with the same reference timestamp to indicate it supports NTPv5.

An NTP client which supports both NTPv4 and NTPv5, and is not configured to use a particular version, SHOULD start with NTPv4 requests having the reference timestamp set to 0x4e5450354e545035. If the server responds with the same reference timestamp, the client SHOULD switch to NTPv5.

## 9. Acknowledgements

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

The author would like to thank Doug Arnold, Watson Ladd, Hal Murray, Kurt Roeckx, and Ulrich Windl for their useful comments.

## 10. IANA Considerations

This memo includes no request to IANA.

## 11. Security Considerations

## 12. References

### 12.1. Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8573] Malhotra, A. and S. Goldberg, "Message Authentication Code for the Network Time Protocol", RFC 8573, DOI 10.17487/RFC8573, June 2019, <https://www.rfc-editor.org/info/rfc8573>.

### 12.2. Informative References

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

Author's Address

Miroslav Lichvar
Red Hat
Purkynova 115
Brno  612 00
Czech Republic

Email: mlichvar@redhat.com