NTP Working Group
Internet Draft
Intended status: Experimental
Expires: January 2015

T. Mizrahi
Marvell

July 21, 2014

# Using UDP Checksum Trailers in the Network Time Protocol (NTP)
## draft-ietf-ntp-checksum-trailer-00.txt

Abstract

The Network Time Protocol (NTP) allows clients to synchronize to a time server using timestamped protocol messages. To faciliate accurate timestamping, some implementations use hardware-based timestamping engines that integrate the accurate transmission time into every outgoing NTP packet during transmission. Since these packets are transported over UDP, the UDP checksum field is then updated to reflect this modification. This document proposes an extension field that includes a 2-octet Checksum Trailer, allowing timestamping engines to reflect the checksum modification in the last 2 octets of the packet rather than in the UDP checksum field. The behavior defined in this document is interoperable with existing NTP implementations.

Table of Contents



## 1. Introduction

The Network Time Protocol [NTPv4] allows clients to synchronize their clocks to a time server by exchanging NTP packets. The increasing demand for highly accurate clock synchronization motivates implementations that provide accurate timestamping.

### 1.1. Intermediate Entities

In this document we use the term 'intermediate entity', referring to an entity that reside on the path between the sender and the receiver of an NTP packet, that modifies this NTP packet en-route. Two examples of intermediate entities are presented below.

In order to facilitate accurate timestamping, an implementation MAY use a hardware based timestamping engine, as shown in Figure 1. In such cases, NTP packets are sent and received by a software layer, whereas a timestamping engine modifies every outgoing NTP packet by incorporating its accurate transmission time into the <Transmit Timestamp> field in the packet.

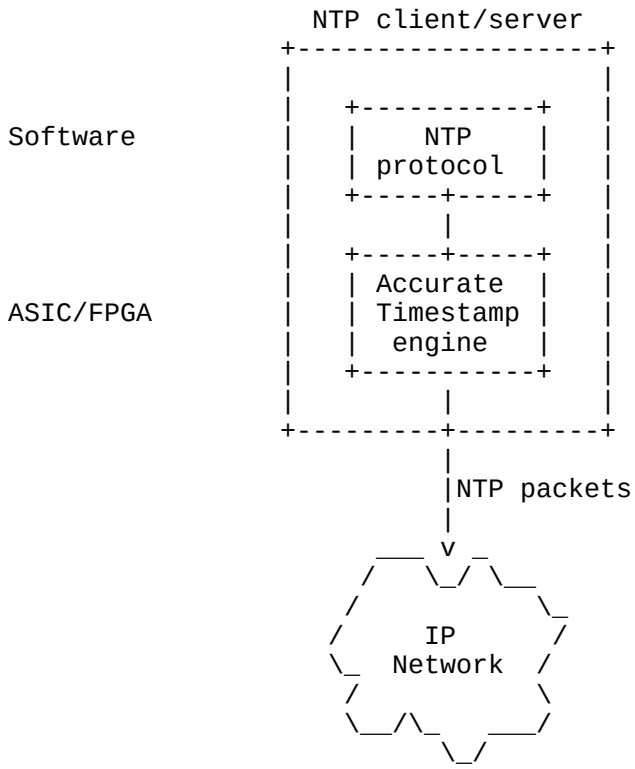



Figure 1 Accurate Timestamping in NTP

The accuracy of clock synchronization over packet networks is highly sensitive to delay jitters in the underlying network, which dramatically affects the clock accuracy. To address this challenge, the Precision Time Protocol (PTP) [IEEE1588] defines Transparent Clocks (TCs), intermediate switches and routers that improve the end-to-end accuracy by updating a "Correction Field" in the PTP packet by adding the latency caused by the current TC. In NTP no equivalent entity is currently defined, but future versions of NTP may define an intermediate node that modifies en-route NTP packets using a "Correction Field".

### 1.2. Updating the UDP Checksum

When the UDP payload is modified by an intermediate entity, the UDP Checksum field needs to be updated to maintain its correctness. When using UDP over IPv4 ([UDP]), an intermediate entity can assign a value of zero in the checksum field, causing the receiver to ignore the checksum field. UDP over IPv6, as defined in [IPv6], does not allow a zero checksum, and requires the UDP checksum field to contain a correct checksum of the UDP payload.

Since an intermediate entity only modifies a specific field in the packet, i.e. the timestamp field, the UDP checksum update can be performed incrementally, using the concepts presented in [Checksum].

A similar problem is addressed in Annex E of [IEEE1588]. When the Precision Time Protocol (PTP) is transported over IPv6, two octets are appended to the end of the PTP payload for UDP checksum updates. The value of these two octets can be updated by an intermediate entity, causing the value of the UDP checksum field to remain correct.

This document defines a similar concept for [NTP], allowing intermediate entities to update NTP packets and maintain the correctness of the UDP checksum by modifying the last 2 octets of the packet. This is performed by adding an NTP extension field at the end of the packet, in which the last two bytes are used as a checksum trailer.

The term Checksum Trailer is used throughout this document and refers to the 2 octets at the end of the UDP payload, used for updating the UDP checksum by intermediate entities.

The usage of the Checksum Trailer can in some cases simplify the implementation, since if the packet data is processed in a serial order, it is simpler to first update the timestamp field, and then update the Checksum Trailer rather than to update the timestamp and then update the UDP checksum, residing at the UDP header.

## 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 [KEYWORDS].

## 2.2. Abbreviations

| | |
|---|---|
| MAC | Message Authentication Code |
| NTP | Network Time Protocol |
| PTP | Precision Time Protocol |
| UDP | User Datagram Protocol |

# 3. Using UDP Checksum Trailers in NTP

## 3.1. Overview

The UDP Checksum Trailer is a two-octet trailer that is appended at the end of the UDP payload using an NTP extension field. Figure 2 illustrates the packet format of an NTP packet with a Checksum Trailer extension. The figure illustrates an unauthenticated NTP packet. Section 3.4. provides further details about using the Checksum Trailer in authenticated packets.

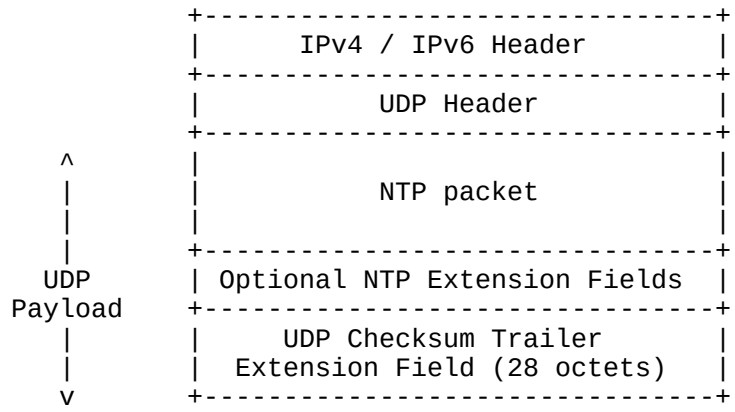



Figure 2 Checksum Trailer in NTP Unauthenticated Packets

## 3.2. Checksum Trailer in NTP Packets

NTP is transported over UDP, either over IPv4 or over IPv6. This document applies to both NTP over IPv4, and NTP over IPv6.

NTP packets may include one or more extension fields, as defined in [NTPv4]. The Checksum Trailer in NTP packets resides in a dedicated NTP extension field, as shown in Figure 2.

In unauthenticated mode, if the NTP packet includes more than one extension field, the Checksum Trailer extension is always the last extension field. Thus, when NTP authentication is disabled, the Checksum Trailer is the last 2 octets in the UDP payload, and thus the trailer is located at UDP Length - 2 octets after the beginning of the UDP header.

When NTP authentication is enabled, the Checksum Trailer is the last 2 octets before the Message Authentication Code (MAC).

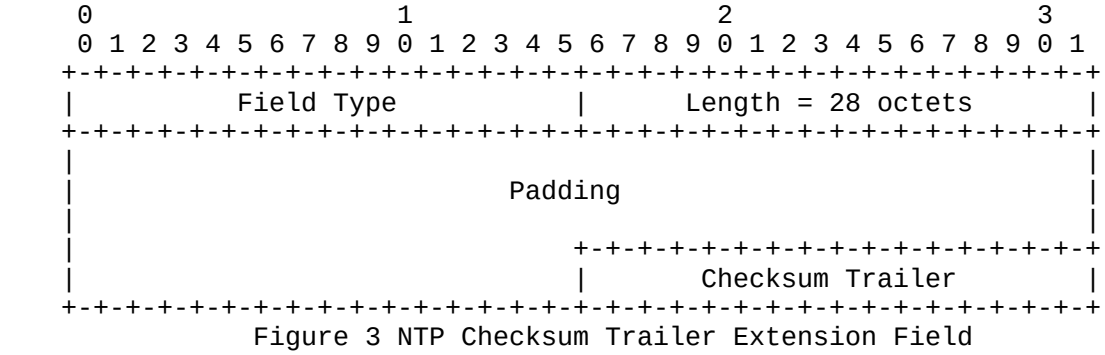



Figure 3 NTP Checksum Trailer Extension Field

Field Type

A dedicated Field Type value is used to identify the Checksum Trailer extension. See Section 6 for further details.

Length

The Checksum Trailer extension field length is 28 octets.

This length guarantees that the host that receives the packet parses it correctly, whether the packet includes a MAC or not. [NTP-Ext] provides further details about the length of an extension field in the absence of a MAC.

Padding

The extension field includes 22 octets of padding. This field SHOULD be set to 0, and SHOULD be ignored by the recipient.

Checksum Trailer

Includes the UDP Checksum Trailer field.

#### 3.2.1. Transmission of NTP with Checksum Trailer

The transmitter of an NTP packet MAY include a Checksum Trailer extension field.

#### 3.2.2. Intermediate Updates of NTP with Checksum Trailer

An intermediate node that receives and alters an NTP packet containing a Checksum Trailer extension MAY use the Checksum Trailer to maintain a correct UDP checksum value.

#### 3.2.3. Reception of NTP with Checksum Trailer

This document does not impose new requirements on the receiving end of an NTP packet.

The UDP layer at the receiving end verifies the UDP Checksum of received NTP packets, and the NTP layer SHOULD ignore the Checksum Trailer extension field.

### 3.3. Interoperability with Existing Implementations

The behavior defined in this document does not impose new requirements on the reception of NTP packets. Thus, transmitters and intermediate nodes that support the Checksum Trailer can transparently interoperate with existing implementations.

### 3.4. Using the Checksum Trailer with or without Authentication

A Checksum Trailer SHOULD NOT be used when authentication is enabled. The Checksum Trailer is effective in unauthenticated mode, allowing the intermediate entity to perform serial processing of the packet without storing-and-forwarding it.

On the other hand, when message authentication is used, an intermediate entity that alters NTP packets must also re-compute the Message Authentication Code (MAC) accordingly. The MAC update typically requires the intermediate entity to store the packet, re-compute its MAC, and then forward it. Thus, the benefit of the checksum trailer is effectively irrelevant when a MAC is used.

## 4. Security Considerations

This document describes how a Checksum Trailer extension can be used for maintaining the correctness of the UDP checksum.

The purpose of this extension is to ease the implementation of accurate timestamping engines, as described in Figure 1. The extension is intended to be used internally in an NTP client or server, and not intended to be used by intermediate switches and routers that reside between the client and the server. As opposed to PTP [IEEE1588], NTP does not require intermediate switches or routers to modify the content of NTP messages, and thus any such modification should be considered as a malicious MITM attack.

It is important to emphasize that the scheme described in this document does not increase the protocol's vulnerabitliy to MITM attacks; a MITM who maliciously modifies a packet and its checksum trailer is logically equivalent to a MITM attacker who modifies a packet and its UDP Checksum field.

The concept described in this document is intended to be used only in unauthenticated mode. As described in Section 3.4. , the benefits of the Checksum Trailer do not apply when authentication is enabled.

## 5. IANA Considerations

IANA is requested to allocate an NTP extension Field Type value for the Checksum Trailer extension.

## 6. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

## 7. References

### 7.1. Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[IPv6] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[Checksum] Rijsinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.

[UDP] Postel, J., "User Datagram Protocol", RFC 768, August 1980.

[NTPv4] Mills, D., Martin, J., Burbank, J., Kasch, W., "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

### 7.2. Informative References

[IEEE1588] IEEE TC 9 Instrumentation and Measurement Society 2000, "1588 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", IEEE Standard, 2008.

[NTP-Ext] Mizrahi, T., Mayer, D., "Using NTP Extension Fields without Authentication", draft-mizrahi-ntp-extension-field (work in progress), January 2014.

## Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

D. Sibold
PTB
S. Roettger
Google Inc
K. Teichel
PTB
R. Housley
Vigil Security
October 23, 2014

# Protecting Network Time Security Messages with the Cryptographic Message Syntax (CMS)

draft-ietf-ntp-cms-for-nts-message-00.txt

## Abstract

This document describes a convention for using the Cryptographic Message Syntax (CMS) to protect the messages in the Network Time Security (NTS) protocol. NTS provides authentication of time servers as well as integrity protection of time synchronization messages using Network Time Protocol (NTP) or Precision Time Protocol (PTP).

## 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].

## Status of This Memo

Table of Contents



## 1. Introduction

This document provides detail on how to construct NTS messages in practice. NTS provides secure time synchronization with time servers using Network Time Protocol (NTP) [RFC5905] or Precision Time Protocol (PTP) [IEEE1588]. Among other things, this document

describes a convention for using the Cryptographic Message Syntax (CMS) [RFC5652] to protect messages in the Network Time Security (NTS) protocol. Encryption is used to provide confidentiality of secrets, and digital signatures are used to provide authentication and integrity of content.

Sometimes CMS is used in an exclusively ASN.1 [ASN1] environment. In this case, the NTS message may use any syntax that facilitates easy implementation.

## 2. CMS Conventions for NTS Message Protection

Regarding the usage of CMS we differentiate between four archetypes according to which the NTS message types can be structured:

NTS-Plain: This archetype is used for actual time synchronization messages (explicitly, the message types: time_request, time_response, server_broad, see [I-D.ietf-ntp-network-time-security], section 6) as well as for the very first messages of a unicast or a broadcast exchange (client_assoc or client_bpar, respectively) and the broadcast keycheck exchange (client_keycheck and server_keycheck). This archetype does not make use of any CMS structures.

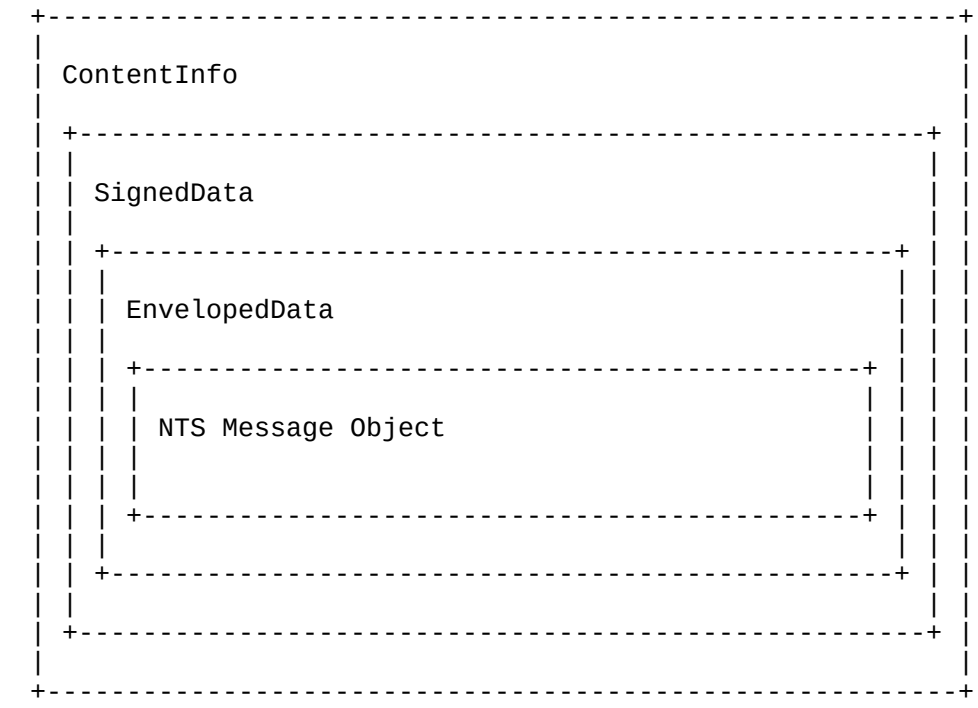
NTS-Signed-and-Encrypted: This archetype is used for secure transmission of the cookie (only for the server_cook message type, see [I-D.ietf-ntp-network-time-security], section 6). For this, the following CMS structure is used:

First, the NTS message MUST be encrypted using the EnvelopedData content type. EnvelopedData supports nearly any form of key management. In the NTS protocol the client provides a certificate in an unprotected message, and the public key from this certificate, if it is valid, will be used to establish a pairwise symmetric key for the encryption of the protected NTS message.

Second, the EnvelopedData content MUST be digitally signed using the SignedData content type. SignedData supports nearly any form of digital signature, and in the NTS protocol the server will include its certificate within the SignedData content type.

Third, the SignedData content type MUST be encapsulated in a ContentInfo content type.

Figure 1 illustrates this structure.

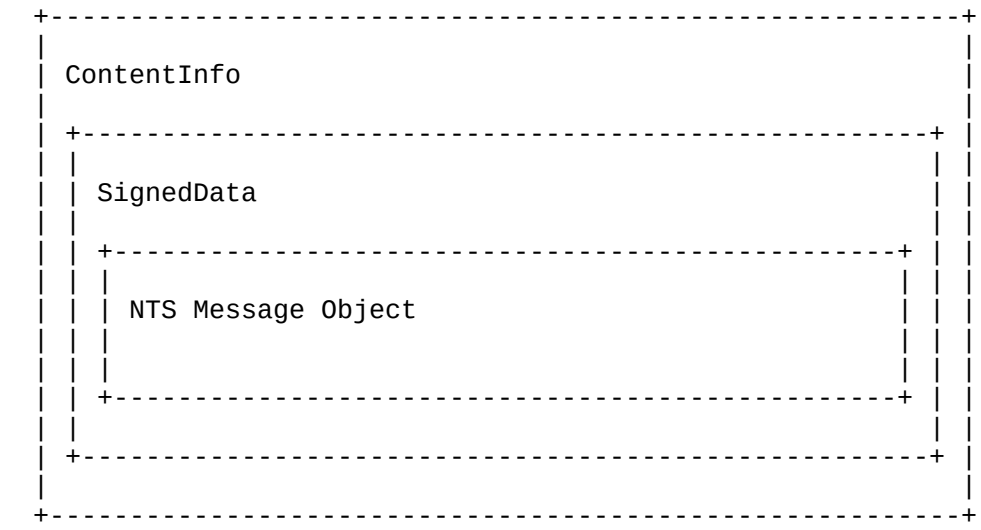



NTS-Signed: This archetype is used for server_assoc and server_bpar message types. It uses the following CMS structure:

First, the NTS message object MUST be wrapped in a SignedData content type. The messages MUST be digitally signed, and certificates included. SignedData supports nearly any form of digital signature, and in the NTS protocol the server will include its certificate within the SignedData content type.

Second, the SignedData content type MUST be encapsulated in a ContentInfo content type.

Figure 2 illustrates this structure.




NTS-Certified: This archetype is used for the client_cook message type. It uses a CMS structure much like the NTS-Signed archetype (see Figure 2), with the only difference being that messages SHOULD NOT be digitally signed. This archetype employs the CMS structure merely in order to transport certificates.

Whichever archetype is used, the resulting structure is always transported in an extension field of an NTP packet. In the case of messages that also need to carry time synchronization data, this data is written into the regular fields of the NTP packet.

## 2.1. Fields of the employed CMS Content Types

Overall, three CMS content types are used for NTS messages: ContentInfo, SignedData and EnvelopedData. The following is a description of how the fields of those content types are used in detail.

### 2.1.1. ContentInfo

The ContentInfo content type is used in all four archetypes. The fields of the SignedData content type are used as follows:

contentType -- indicates the type of the associated content. For the archetype NTS-Plain, it MUST identify the NTS message object that is included. For all other archetypes (NTS-Certified, NTS-Signed and NTS-Signed-and-Encrypted), it MUST contain the object identifier for the SignedData content type:

```
id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

content is the associated content. For the NTS-Plain archetype, it MUST contain the DER encoded NTS message object. For all other archetypes, it MUST contain the DER encoded SignedData content type.

### 2.1.2. SignedData

The SignedData content type is used in the NTS-Certified, NTS-Signed and NTS-Signed-and-Encrypted archetypes but not in the NTS-Plain archetype. The fields of the SignedData content type are used as follows:

version -- the appropriate value depends on the optional items that are included. In the NTS protocol, the signer certificate MUST be included, and other items MAY be included. The instructions in [RFC5652] section 5.1 MUST be followed to set the correct value.

digestAlgorithms -- is a collection of message digest algorithm identifiers. In the NTS protocol, there MUST be exactly one algorithm identifier present. The instructions in Section 5.4 of [RFC5652] MUST be followed.

encapContentInfo -- this structure is always present. In the NTS protocol, it MUST follow these conventions:

eContentType -- is an object identifier. In the NTS protocol, for the NTS-Certified and NTS-Signed archetypes, it MUST identify the type of the NTS message that was encapsulated. For the NTS-Signed-and-Encrypted archetype, it MUST contain the object identifier for the EnvelopedData content type:

```
id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
       us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }.
```

eContent is the content itself, carried as an octet string. For the NTS-Certified and NTS-Signed archetypes, it MUST contain the DER encoded encapsulated NTS message object. The instructions in Section 6.3 of [RFC5652] MUST be followed. For the NTS-Signed-and-Encrypted archetype, it MUST contain the DER encoded EnvelopedData content type.

certificates -- is a collection of certificates. In the NTS protocol, it MUST contain the DER encoded certificate [RFC5280] of the sender. It is intended that the collection of certificates be

sufficient for the recipient to construct a certification path from a recognized "root" or "top-level certification authority" to the certificate used by the sender.

crls -- is a collection of revocation status information. In the NTS protocol, it MAY contain one or more DER encoded CRLs [RFC5280]. It is intended that the collection contain information sufficient to determine whether the certificates in the certificates field are valid.

signerInfos -- is a collection of per-signer information. In the NTS protocol, for the NTS-Certified archetype, this SHOULD be left out. For both the NTS-Signed and the NTS-Signed-and-Encrypted archetypes, there MUST be exactly one SignerInfo structure present. The details of the SignerInfo type are discussed in Section 5.3 of [RFC5652]. In the NTS protocol, it MUST follow these conventions:

version -- is the syntax version number. In the NTS protocol, the SignerIdentifier is subjectKeyIdentifier, therefore the version MUST be 3.

sid -- identifies the signer’s certificate. In the NTS protocol, the sid field contains the subjectKeyIdentifier from the signer’s certificate.

digestAlgorithm -- identifies the message digest algorithm, and any associated parameters, used by the signer. In the NTS protocol, the identifier MUST match the single algorithm identifier present in the digestAlgorithms.

signedAttrs -- is a collection of attributes that are signed. In the NTS protocol, it MUST be present, and it MUST contain the following attributes:

Content Type -- see Section 11.1 of [RFC5652].

Message Digest -- see Section 11.2 of [RFC5652].

In addition, it MAY contain the following attributes:

Signing Time -- see Section 11.3 of [RFC5652].

Binary Signing Time -- see Section 3 of [RFC5652].

signatureAlgorithm -- identifies the signature algorithm, and any associated parameters, used by the signer to generate the digital signature.

signature is the result of digital signature generation, using the message digest and the signer's private key. The instructions in Section 5.5 of [RFC5652] MUST be followed.

unsignedAttrs -- is an optional collection of attributes that are not signed. In the NTS protocol, the it MUST be absent.

### 2.1.3. EnvelopedData

The EnvelopedData content type is used only in the NTS-Signed-and-Encrypted archetype. The fields of the EnvelopedData content type are used as follows:

version -- the appropriate value depends on the type of key management that is used. The instructions in [RFC5652] section 6.1 MUST be followed to set the correct value.

originatorInfo -- this structure is present only if required by the key management algorithm. In the NTS protocol, it MUST be present when a key agreement algorithm is used, and it MUST absent when a key transport algorithm is used. The instructions in Section 6.1 of [RFC5652] MUST be followed.

recipientInfos -- this structure is always present. In the NTS protocol, it MUST contain exactly one entry that allows the client to determine the key used to encrypt the NTS message. The instructions in Section 6.2 of [RFC5652] MUST be followed.

encryptedContentInfo -- this structure is always present. In the NTS protocol, it MUST follow these conventions:

contentType -- indicates the type of content. In the NTS protocol, it MUST identify the type of the NTS message that was encrypted.

contentEncryptionAlgorithm -- identifies the content-encryption algorithm, and any associated parameters, used to encrypt the content.

encryptedContent -- is the encrypted the content. In the NTS protocol, it MUST contain the encrypted NTS message. The instructions in Section 6.3 of [RFC5652] MUST be followed.

unprotectedAttrs -- this structure is optional. In the NTS protocol, it MUST be absent.

## 3. Certificate Conventions

The syntax and processing rules for certificates are specified in [RFC5652]. In the NTS protocol, the server certificate MUST contain the following extensions:

Subject Key Identifier -- see Section 4.2.1.2 of [RFC5652].

Key Usage -- see Section 4.2.1.3 of [RFC5652].

Extended Key Usage -- see Section 4.2.1.22 of [RFC5652].

The Extended Key Usage extension MUST include the id-kp-NTSserver object identifier. When a certificate issuer includes this object identifier in the extended key usage extension, it provides an attestation that the certificate subject is a time server that supports the NTS protocol.

The id-kp-NTSserver object identifier is:

```
id-kp-NTSserver OBJECT IDENTIFIER ::= { TBD }
```

## 4. Implementation Notes: ASN.1 Structures and Use of the CMS

This section gives some hints on the structures of the NTS message objects for the different message types when one wishes to implement the protocol.

### 4.1. Preliminaries

The following ASN.1 coded data type "NTSNonce" is needed for other types used below for NTS messages. It specifies a 128 bit nonce as required in several message types:

```
NTSNonce ::= OCTET STRING (SIZE(16))
```

### 4.2. Unicast Messages

#### 4.2.1. Association Messages

##### 4.2.1.1. Message Type: "client_assoc"

This message is structured according to the NTS-Plain archetype. It is realized as an NTP packet with an extension field which holds all the data relevant for NTS. Explicitly, the extension field contains an ASN.1 object of type "ClientAssocData", which is structured as follows:

```
ClientAssocData ::= SEQUENCE {
    clientId         SubjectKeyIdentifier,
    digestAlgos      DigestAlgorithmIdentifiers,
    keyEncAlgos      KeyEncryptionAlgorithms,
    contentEncAlgos  ContentEncryptionAlgorithms
}
```

#### 4.2.1.2. Message Type: "server_assoc"

This message is structured according to the NTS-Signed archetype. The NTS message object in this case is an ASN.1 object of type "ServerAssocData", which is structured as follows:

```
ServerAssocData ::= SEQUENCE {
    clientId              SubjectKeyIdentifier,
    choiceDigestAlgo      DigestAlgorithmIdentifier,
    choiceKeyEncAlgo      KeyEncryptionAlgorithmIdentifier,
    choiceContentEncAlgo  ContentEncryptionAlgorithmIdentifier
}
```

### 4.2.2. Cookie Messages

#### 4.2.2.1. Message Type: "client_cook"

This message is structured according to the NTS-Certified archetype. The NTS message object is a "ClientCookieData" type ASN.1 object, structured as follows:

```
ClientCookieData ::= SEQUENCE {
    nonce       NTSNonce,
    signAlgo    SignatureAlgorithmIdentifier,
    digestAlgo  DigestAlgorithmIdentifier,
    encAlgo     ContentEncryptionAlgorithmIdentifier,
    keyEncAlgo  KeyEncryptionAlgorithmIdentifier
}
```

It is identified by the following object identifier (fictional values):

```
id-clientCookieData OBJECT IDENTIFIER ::=
    {nts(??) cookie(3) clientcookiedata(1)}
```

#### 4.2.2.2. Message Type: "server_cook"

This message is structured according to the "NTS-Signed-and-Encrypted" archetype. The NTS message object is a "ServerCookieData" object, specified as:

```
ServerCookieData ::= SEQUENCE {
    nonce     NTSNonce,
    cookie    OCTET STRING (SIZE(16))
}
```

It is identified by the following object identifier (fictional values):

```
id-serverCookieData OBJECT IDENTIFIER ::=
    {nts(??) cookie(3) servercookiedata(2)}
```

#### 4.2.3. Time Synchronization Messages

##### 4.2.3.1. Message Type: "time_request"

This message is structured according to the "NTS-Plain" archetype. It is realized as an NTP packet which actually contains regular NTP time synchronization data, as an unsecured NTP packet from a client to a server would. Furthermore, the packet has an extension field which contains an ASN.1 object of type "TimeRequestSecurityData", whose structure is as follows:

```
TimeRequestSecurityData ::=
SEQUENCE {
    nonce_t           NTSNonce,
    digestAlgo        DigestAlgorithmIdentifier,
    hashOfClientCert  BIT STRING
}
```

##### 4.2.3.2. Message Type: "time_response"

This message is also structured according to "NTS-Plain". It is realized as an NTP packet which, like "time_request", contains regular NTP time synchronization data, as an unsecured NTP packet from a server back to a client would. The packet also has an extension field which contains an ASN.1 object of type "TimeResponseSecurityData", with the following structure:

```
TimeResponseSecurityData ::=
SEQUENCE {
    nonce_t  NTSNonce,
}
```

Finally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

### 4.3. Broadcast Messages

#### 4.3.1. Broadcast Parameter Messages

##### 4.3.1.1. Message Type: "client_bpar"

This first broadcast message is structured according to the NTS-Plain archetype. It is realized as an NTP packet which is empty except for an extension field which contains ans ASN.1 object of type "BroadcastParameterRequest", which is structured as follows:

```
BroadcastParameterRequest ::=
SEQUENCE {
    clientId  SubjectKeyIdentifier
}
```

##### 4.3.1.2. Message Type: "server_bpar"

This message is structured according to "NTS-Signed". It is realized as an NTP packet whose extension field carries the necessary CMS structure. The NTS message object in this case is an ASN.1 object of type "BroadcastParameterResponse", with the following structure:

```
BroadcastParameterRequest ::=
SEQUENCE {
    oneWayAlgo1         DigestAlgorithmIdentifier,
    oneWayAlgo2         DigestAlgorithmIdentifier,
    lastKey             OCTET STRING (SIZE (16)),
    intervalDuration    BIT STRING,
    disclosureDelay     INTEGER,
    nextIntervalTime    BIT STRING,
    nextIntervalIndex   INTEGER
}
```

#### 4.3.2. Broadcast Time Synchronization Message

##### 4.3.2.1. Message Type: "server_broad"

This message is structured according to the "NTS-Plain" archetype. Its realization works via an NTP packet which carries regular NTP broadcast time data as well as an extension field, which contains an ASN.1 object of type "BroadcastTime". It has the following structure:

```
BroadcastTime ::=
SEQUENCE {
    thisIntervalIndex   INTEGER,
    disclosedKey        OCTET STRING (SIZE (16)),
}
```

In addition, this packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

#### 4.3.3. Broadcast Keycheck

##### 4.3.3.1. Message Type: "client_keycheck"

This message is structured according to the "NTS-Plain" archetype. It is realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ClientKeyCheckSecurityData", whose structure is as follows:

```
ClientKeyCheckSecurityData ::=
SEQUENCE {
    nonce_k           NTSNonce,
    interval_number   INTEGER,
    digestAlgo        DigestAlgorithmIdentifier,
    hashOfClientCert  BIT STRING
}
```

##### 4.3.3.2. Message Type: "server_keycheck"

This message is also structured according to "NTS-Plain". It is also realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ServerKeyCheckSecurityData", with the following structure:

```
ServerKeyCheckSecurityData ::=
SEQUENCE {
    nonce_t           NTSNonce,
    interval_number   INTEGER
}
```

Additionally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

## 5. IANA Considerations

IANA needs to assign an object identifier for id-kp-NTSserver key purpose and another one for the ASN.1 module in the appendix.

## 6. Security Considerations

To be written.

## 7. References

### 7.1. Normative References

[ASN1] International Telecommunication Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, November 2008.

[IEEE1588] IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems", 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

### 7.2. Informative References

[I-D.ietf-ntp-network-time-security] Sibold, D., Roettger, S., and K. Teichel, "Network Time Security", draft-ietf-ntp-network-time-security-04 (work in progress), July 2014.

## Appendix A. ASN.1 Module

The ASN.1 module contained in this appendix defines the id-kp-NTSserver object identifier.

```
      NTSserverKeyPurpose
        { TBD }

      DEFINITIONS IMPLICIT TAGS ::=
      BEGIN

      id-kp-NTSserver OBJECT IDENTIFIER ::= { TBD }

      END
```

## Authors' Addresses

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

Stephen Roettger
Google Inc

Email: stephen.roettger@googlemail.com

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

Phone: +49-(0)531-592-8421
Email: kristof.teichel@ptb.de

Russ Housley
Vigil Security

NTP Working Group
Internet Draft
Intended status: Standards Track
Updates: 5905
Expires: December 2014

T. Mizrahi
Marvell
D. Mayer
Network Time Foundation

June 26, 2014

# Using NTP Extension Fields without Authentication
## draft-ietf-ntp-extension-field-01.txt

Abstract

The Network Time Protocol Version 4 (NTPv4) defines the optional usage of extension fields. An extension field is an optional field that resides at the end of the NTP header, and can be used to add optional capabilities or additional information that is not conveyed in the standard NTP header. The current definition of extension fields in NTPv4 is somewhat ambiguous regarding the connection between extension fields and the presence of a Message Authentication Code (MAC). This draft clarifies the usage of extension fields in the presence and in the absence of a MAC, while maintaining interoperability with existing implementations.

Table of Contents

## 1. Introduction

The NTP header format consists of a set of fixed fields that may be followed by some optional fields. Two types of optional fields are defined, Message Authentication Codes (MAC), and extension fields.

If a MAC is used, it resides at the end of the packet. This field can be either 24 octets long, 20 octets long, or a 4-octet crypto-NAK.

NTP extension fields were defined in [RFC5905] as a generic mechanism that allows to add future extensions and features without modifying the NTP header format.

The only currently defined extension field is the one used by the AutoKey protocol [RFC5906].

The NTP specification is somewhat ambiguous with regards to the connection between using extension fields and the presence of a MAC.

- The definition of the NTP extension field implies that it was intended to be a generic mechanism that can be used for various future features of the protocol (see Section A.1.).

- On the other hand, the NTP extension field description in [RFC5905] states that a MAC is always present when an extension field is present (see Section A.2.).

The last two quotes seem to be in contradiction; since the extension field was defined as a generic future-compatible building block, it seems unlikely to bind it to a specific feature in the protocol.

Moreover, the extension field parsing rules presented in [RFC5906] imply that an extension field can be present without a MAC, provided that the extension field is at least 28 Octets long.

This document attempts to resolve the ambiguity with regards to the connection between NTP extension fields and MACs, updating Section 7.5 of [RFC5905], and describes the usage of extension fields in the absence of a MAC in a way that is interoperable with current implementations.

## 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 [KEYWORDS].

### 2.2. Terms & Abbreviations

| | |
|---|---|
| NTPv4 | Network Time Protocol Version 4 |
| MAC | Message Authentication Code |

## 3. NTP Extension Fields with and without a MAC - Clarifications

This section clarifies the usage of extension fields in the absence of a MAC, in accordance with the definitions in [RFC5905] and [RFC5906]. Section 4. defines a more generic and flexible usage of extension fields.

### 3.1. Extension Field Format

The NTP extension field is defined in Section 7.5 of [RFC5905]. The extension field format is quoted here in Section A.3.

The minimal length of an extension field, as defined in Section 7.5 of [RFC5905], is 16 octets.

### 3.2. Extension Fields in the Absence of a MAC

Extension fields can be used when a MAC is not present in the NTP packet. In this case, the extension fields must comply with the parsing rules in Section A.4. Specifically:

- If the packet includes a single extension field, the length of the extension field MUST be at least 7 words, i.e., at least 28 octets.

- If the packet includes more than one extension field, the length of the last extension field MUST be at least 28 octets. The length of the other extension fields in this case MUST be at least 16 octets each, as defined in [RFC5905].

A host that supports NTP extension fields MUST parse NTP extension fields as described in Section A.4.

### 3.3. Unknown Extension Fields

If an extension field is unknown to the receiving server the server should ignore the extension field and may optionally drop the packet altogether if policy requires it. Note that in the presence of an unknown extension field any MAC that may be present may be misinterpreted as an unknown extension though in this case the apparent extension length will be totally inconsistent with the total length of the rest of the packet.

### 3.4. Interoperability with Current Implementations

The behavior described in Section 3.2. is compliant to [RFC5906], and thus should be compatible with existing implementations that support NTP extension fields.

## 4. NTP Extension Field Usage with and without a MAC - Extensions

This section updates [RFC5905] and [RFC5906] with respect to the usage of extension fields, allowing a more flexible and unambiguous usage.

### 4.1. Extension Fields in the Presence of a MAC

The usage of extension fields in the presence of a MAC is specified in [RFC5905] and in [RFC5906]. The requirement for a MAC MUST be specified by the specification for the extension field and the specification MUST include both the algorithm to be used to create the MAC and the length of the MAC thus created. An extension field may allow for more than one algorithm to be used in which case the information about which one was used MUST be included in the extension field itself.

### 4.2. Extension Fields in the Absence of a MAC

Extension fields can be used when a MAC is not present in the NTP packet. In this case, the extension fields must comply with the following:

- If the packet includes a single extension field, the length of the extension field MUST be at least 16 octets. The extension length is specified in the length field of the extension and is the number of octets in the extension field.

- o If the packet includes more than one extension field, the length of the last extension field MUST be at least 28 octets. The length of the other extension fields in this case MUST be at least 16 octets each, as defined in [RFC5905].

### 4.3. Multiple Extension fields in an NTP packet

If there are multiple extension fields that require a MAC they MUST all require use of the same algorithm and MAC length. Extension fields that do not require a MAC can be included with extension fields that do require a MAC.

### 4.4. MAC in the absence of an Extension field

A MAC must not be any longer than 24 octets if there is no extension field present unless through a previous exchange of packets with an extension field which defines the size and algorithm of the MAC transmitted in the packet and is agreed upon by both client and server.

## 5. Security Considerations

The security considerations of the network time protocol are discussed in [RFC5905]. This document clarifies some ambiguity with regards to the usage of the NTP extension field, and thus the behavior described in this document does not introduce new security considerations.

## 6. IANA Considerations

There are no new IANA considerations implied by this document.

## 7. Acknowledgments

The authors thank Dave Mills for his insightful comments.

This document was prepared using 2-Word-v2.0.template.dot.

## 8. References

### 8.1. Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5905] Mills, D., Martin, J., Burbank, J., Kasch, W., "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

### 8.2. Informative References

[RFC5906] Haberman, B., Mills, D., "Network Time Protocol Version 4: Autokey Specification", RFC 5906, June 2010.

## Appendix A. Requirements from NTPv4 and Autokey

### A.1. NTP Extension Field for Future Extensions

The following paragraph is quoted from Section 16 of [RFC5905].

> This document introduces NTP extension fields allowing for the development of future extensions to the protocol, where a particular extension is to be identified by the Field Type sub-field within the extension field.

### A.2. NTP Extension Field in the Presence of a MAC

The following paragraph is quoted from Section 7.5 of [RFC5905].

> 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.

### A.3. The NTP Extension Field Format

Figure 1 specifies the NTP extension field format, and is quoted from [RFC5905]. For further details refer to [RFC5905].

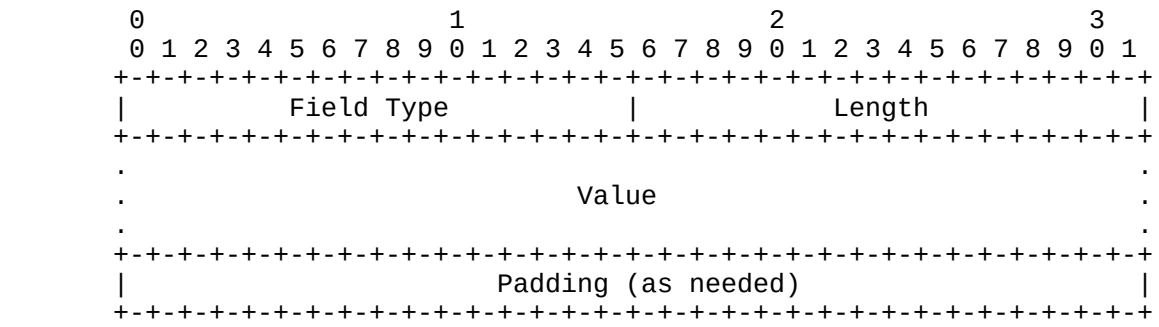



Figure 1 The NTP Extension Field Format

## A.4. NTP Extension Field in Autokey

The following paragraph is quoted from Section 10 of [RFC5906].

One or more extension fields follow the NTP packet header and the last followed by the MAC. The extension field parser initializes a pointer to the first octet beyond the NTP packet header and calculates the number of octets remaining to the end of the packet If the remaining length is 20 (128-bit digest plus 4-octet key ID) or 22 (160-bit digest plus 4-octet key ID), the remaining data are the MAC and parsing is complete. If the remaining length is greater than 22, an extension field is present. If the remaining length is less than 8 or not a multiple of 4, a format error has occurred and the packet is discarded; otherwise, the parser increments the pointer by the extension field length and then uses the same rules as above to determine whether a MAC is present or another extension field.

## Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

Danny Mayer
Network Time Foundation
PO Box 918
Talent OR 97540

Email: mayer@ntp.org

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

D. Sibold
PTB
S. Roettger
Google Inc
K. Teichel
PTB

October 23, 2014

# Network Time Security
# draft-ietf-ntp-network-time-security-05.txt

## Abstract

This document describes the Network Time Security (NTS) protocol that enables secure time synchronization with time servers using Network Time Protocol (NTP) or Precision Time Protocol (PTP). Its design considers the special requirements of precise timekeeping, which are described in Security Requirements of Time Protocols in Packet Switched Networks [RFC7384].

## 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].

## 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 http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on April 26, 2015.

Table of Contents

## 1. Introduction

Time synchronization protocols are increasingly utilized to synchronize clocks in networked infrastructures. The reliable performance of such infrastructures can be degraded seriously by successful attacks against the time synchronization protocol. Therefore, time synchronization protocols have to be secured if they are applied in environments that are prone to malicious attacks. This can be accomplished by utilization of external security protocols like IPsec or by intrinsic security measures of the time synchronization protocol.

The two most popular time synchronization protocols, the Network Time Protocol (NTP) [RFC5905] and the Precision Time Protocol (PTP) [IEEE1588], currently do not provide adequate intrinsic security precautions. This document specifies security measures for NTP and PTP which enable these protocols to verify authenticity of the time server and integrity of the time synchronization protocol packets.

The protocol is specified with the prerequisite in mind that precise timekeeping can only be accomplished with stateless time synchronization communication, which excludes the utilization of standard security protocols like IPsec or TLS for time synchronization messages. This prerequisite corresponds with the requirement that a security mechanism for timekeeping must be designed in such a way that it does not degrade the quality of the time transfer [RFC7384].

Note:

The intent is to formulate the protocol to be applicable to NTP and also PTP. In the current state the specification focuses on the application to NTP.

## 2. Security Threats

A profound analysis of security threats and requirements for NTP and PTP can be found in the "Security Requirements of Time Protocols in Packet Switched Networks" [RFC7384].

## 3. Objectives

The objectives of the NTS specification are as follows:

- Authenticity: NTS enables the client to authenticate its time servers.
- Integrity: NTS protects the integrity of time synchronization protocol packets via a message authentication code (MAC).
- Confidentiality: NTS does not provide confidentiality protection of the time synchronization packets.
- Modes of operation: All operational modes of NTP are supported.
- Operational modes of PTP should be supported as far as possible.
- Hybrid mode: Both secure and insecure communication modes are possible for NTP servers and clients, respectively.
- Compatibility:
  - Unsecured NTP associations shall not be affected.
  - An NTP server that does not support NTS shall not be affected by NTS authentication requests.

## 4. Terms and Abbreviations

MITM Man In The Middle

NTP Network Time Protocol [RFC5905]

NTS Network Time Security

PTP Precision Time Protocol [IEEE1588]

TESLA Timed Efficient Stream Loss-Tolerant Authentication

## 5. NTS Overview

### 5.1. Symmetric and Client/Server Mode

NTS applies X.509 certificates to verify the authenticity of the time server and to exchange a symmetric key, the so-called cookie. This cookie is then used to protect authenticity and integrity of the subsequent time synchronization packets by means of a Message Authentication Code (MAC), which is attached to each time synchronization packet. The calculation of the MAC includes the whole time synchronization packet and the cookie which is shared between client and server. The cookie is calculated according to:

cookie = MSB_128 (HMAC(server seed, H(certificate of client))),

with the server seed as key, where H is a hash function, and where the function MSB_128 cuts off the 128 most significant bits of the result of the HMAC function. The server seed is a 128 bit random value of the server, which has to be kept secret. The cookie never changes as long as the server seed stays the same, but the server seed has to be refreshed periodically in order to provide key freshness as required in [RFC7384]. See Section 8 for details on the seed refresh and Section 7.1.1 for the client’s reaction to it.

The server does not keep a state of the client. Therefore it has to recalculate the cookie each time it receives a request from the client. To this end, the client has to attach the hash value of its certificate to each request (see Section 6.3).

### 5.2. Broadcast Mode

Just as in the case of the client server mode and symmetric mode, authenticity and integrity of the NTP packets are ensured by a MAC, which is attached to the NTP packet by the sender. Verification of the packets’ authenticity is based on the TESLA protocol, in particular on its "not re-using keys" scheme, see section 3.7.2 of

[RFC4082]. TESLA uses a one-way chain of keys, where each key is the output of a one-way function applied to the previous key in the chain. The last element of the chain is shared securely with all clients. The server splits time into intervals of uniform duration and assigns each key to an interval in reverse order, starting with the penultimate. At each time interval, the server sends an NTP broadcast packet appended by a MAC, calculated using the corresponding key, and the key of the previous disclosure interval. The client verifies the MAC by buffering the packet until the disclosure of the key in its associated disclosure interval. In order to be able to verify the validity of the key, the client has to be loosely time synchronized to the server. This has to be accomplished during the initial client server exchange between broadcast client and server. In addition, NTS uses another, more rigorous check to what is used in the TESLA protocol. For a more detailed description of how NTS employs and customizes TESLA, see Appendix C.

## 6. Protocol Messages

This section describes the types of messages needed for secure time synchronization with NTS.

For some guidance on how these message types can be realized in practice, for use with existing time synchronization protocols, see [I-D.ietf-ntp-cms-for-nts-messages], a companion document for NTS. Said document describes ASN.1 encodings for those message parts that have to be added to a time synchronization protocol for security reasons as well as CMS (Cryptographic Message Syntax, see [RFC5652]) conventions that can be used to get the cryptographic aspects right.

Note that currently, the companion document describes realizations of NTS messages only for utilization with NTP, in which the NTS specific data are enclosed in extension fields on top of NTP packets. A specification of NTS messages for PTP will have to be developed accordingly.

The steps described in Section 6.1 - Section 6.3 belong to the unicast mode, while Section 6.4 and Section 6.5 explain the steps involved in the broadcast mode of NTS.

### 6.1. Association Messages

In this message exchange, the hash and encryption algorithms that are used throughout the protocol are negotiated. Also, the client receives the certification chain up to a trusted anchor. With the established certification chain the client is able to verify the

server’s signatures and, hence, authenticity of future NTS messages from the server is ensured.

#### 6.1.1. Message Type: "client_assoc"

The protocol sequence starts with the client sending an association message, called client_assoc. This message contains

- the NTS message ID "client_assoc",
- the version number of NTS that the client wants to use (this SHOULD be the highest version number that it supports),
- the hostname of the client,
- a selection of accepted hash algorithms, and
- a selection of accepted encryption algorithms.

#### 6.1.2. Message Type: "server_assoc"

This message is sent by the server upon receipt of client_assoc. It contains

- the NTS message ID "server_assoc",
- the version number used for the rest of the protocol (which SHOULD be determined as the minimum over the client’s suggestion in the client_assoc message and the highest supported by the server),
- the hostname of the server, and
- the server’s choice of algorithm for encryption and for cryptographic hashing, all of which MUST be chosen from the client’s proposals.
- a signature, calculated over the data listed above, with the server’s private key and according to the signature algorithm which is also used for the certificates which are included (see below),
- a chain of certificates, which starts at the server and goes up to a trusted authority, and each certificate MUST be certified by the one directly following it.

### 6.2. Cookie Messages

During this message exchange, the server transmits a secret cookie to the client securely. The cookie will be used for integrity protection during unicast time synchronization.

#### 6.2.1. Message Type: "client_cook"

This message is sent by the client, upon successful authentication of the server. In this message, the client requests a cookie from the server. The message contains

- the NTS message ID "client_cook",
- the negotiated version number,
- the negotiated signature algorithm,
- the negotiated encryption algorithm,
- a 128-bit nonce,
- the negotiated hash algorithm H,
- the client’s certificate.

#### 6.2.2. Message Type: "server_cook"

This message is sent by the server, upon receipt of a client_cook message. The server generates the hash of the client’s certificate, as conveyed during client_cook, in order to calculate the cookie according to Section 5.1. This message contains

- the NTS message ID "server_cook"
- the version number as transmitted in client_cook,
- a concatenated datum, which is encrypted with the client’s public key, according to the encryption algorithm transmitted in the client_cook message. The concatenated datum contains
  - the nonce transmitted in client_cook, and
  - the cookie.
- a signature, created with the server’s private key, calculated over all of the data listed above. This signature MUST be

calculated according to the transmitted signature algorithm from the client_cook message.

### 6.3. Unicast Time Synchronisation Messages

In this message exchange, the usual time synchronization process is executed, with the addition of integrity protection for all messages that the server sends. This message can be repeatedly exchanged as often as the client desires and as long as the integrity of the server's time responses is verified successfully.

#### 6.3.1. Message Type: "time_request"

This message is sent by the client when it requests time exchange. It contains

- the NTS message ID "time_request",
- the negotiated version number,
- a 128-bit nonce,
- the negotiated hash algorithm H,
- the hash of the client's certificate under H.

#### 6.3.2. Message Type: "time_response"

This message is sent by the server, after it received a time_request message. Prior to this the server MUST recalculate the client's cookie by using the hash of the client's certificate and the transmitted hash algorithm. The message contains

- the NTS message ID "time_response",
- the version number as transmitted in time_request,
- the server's time synchronization response data,
- the 128-bit nonce transmitted in time_request,
- a MAC (generated with the cookie as key) for verification of all of the above data.

### 6.4. Broadcast Parameter Messages

In this message exchange, the client receives the necessary information to execute the TESLA protocol in a secured broadcast association. The client can only initiate a secure broadcast association after a successful unicast run, see Section 7.1.2.

See Appendix C for more details on TESLA.

#### 6.4.1. Message Type: "client_bpar"

This message is sent by the client in order to establish a secured time broadcast association with the server. It contains

- the NTS message ID "client_bpar",
- the version number negotiated during association in unicast mode,
- the client's hostname, and
- the signature algorithm negotiated during unicast.

#### 6.4.2. Message Type: "server_bpar"

This message is sent by the server upon receipt of a client_bpar message during the broadcast loop of the server. It contains

- the NTS message ID "server_bpar",
- the version number as transmitted in the client_bpar message,
- the one-way functions used for building the key chain, and
- the disclosure schedule of the keys. This contains:
  - the last key of the key chain,
  - time interval duration,
  - the disclosure delay (number of intervals between use and disclosure of a key),
  - the time at which the next time interval will start, and
  - the next interval's associated index.
- The message also contains a signature signed by the server with its private key, verifying all the data listed above.

### 6.5. Broadcast Messages

Via this message, the server keeps sending broadcast time synchronization messages to all participating clients.

#### 6.5.1. Message Type: "server_broad"

This message is sent by the server over the course of its broadcast schedule. It is part of any broadcast association. It contains

- the NTS message ID "server_broad",
- the version number that the server’s broadcast mode is working under,
- time broadcast data,
- the index that belongs to the current interval (and therefore identifies the current, yet undisclosed key),
- the disclosed key of the previous disclosure interval (current time interval minus disclosure delay),
- a MAC, calculated with the key for the current time interval, verifying
  - the message ID,
  - the version number, and
  - the time data.

### 6.6. Broadcast Key Check

This message exchange is performed for an additional check of packet timeliness in the course of the TESLA scheme, see Appendix C.

#### 6.6.1. Message Type: "client_keycheck"

A message of this type is sent by the client in order to initiate an additional check of packet timeliness for the TESLA scheme. It contains

- the NTS message ID "client_keycheck",
- the version number chosen for the broadcast,
- a 128-bit nonce,

- an interval number from the TESLA disclosure schedule,
- the hash algorithm H negotiated in unicast mode, and
- the hash of the client's certificate under H.

#### 6.6.2. Message Type: "server_keycheck"

A message of this type is sent by the server upon receipt of a client_keycheck message during the broadcast loop of the server. Prior to this the server MUST recalculate the client's cookie by using the hash of the client's certificate and the transmitted hash algorithm. It contains

- the NTS message ID "server_keycheck"
- the version number that the server's broadcast mode is working under,
- the 128-bit nonce transmitted in the client_keycheck message,
- the interval number transmitted in the client_keycheck message, and
- a MAC (generated with the cookie as key) for verification of all of the above data.

## 7. Protocol Sequence

### 7.1. The Client

#### 7.1.1. The Client in Unicast Mode

For a unicast run, the client performs the following steps:

1. It sends a client_assoc message to the server. It MUST keep the transmitted values for version number and algorithms available for later checks.
2. It waits for a reply in the form of a server_assoc message. After receipt of the message it performs the following checks:
   * The client checks that the message contains a conform version number.
   * It also verifies that the server has chosen the encryption and hash algorithms from its proposal sent in the client_assoc message.

* Furthermore, it performs authenticity checks on the certificate chain and the signature for the version number.

If one of the checks fails, the client MUST abort the run.
Discussion:

Note that by performing the above message exchange and checks, the client validates the authenticity of its immediate NTP server only. It does not recursively validate the authenticity of each NTP server on the time synchronization chain. Recursive authentication (and authorization) as formulated in [RFC7384] depends on the chosen trust anchor.

3. Next, it sends a client_cook message to the server. The client MUST save the included nonce until the reply has been processed.

4. It awaits a reply in the form of a server_cook message; upon receipt it executes the following actions:

   * It verifies that the received version number matches the one negotiated before.

   * It verifies the signature using the server’s public key. The signature has to authenticate the encrypted data.

   * It decrypts the encrypted data with its own private key.

   * It checks that the decrypted message is of the expected format: the concatenation of a 128 bit nonce and a 128 bit cookie.

   * It verifies that the received nonce matches the nonce sent in the client_cook message.

   If one of those checks fails, the client MUST abort the run.

5. The client sends a time_request message to the server. The client MUST save the included nonce and the transmit_timestamp (from the time synchronization data) as a correlated pair for later verification steps.

6. It awaits a reply in the form of a time_response message. Upon receipt, it checks:

   * that the transmitted version number matches the one negotiated before,

* that the transmitted nonce belongs to a previous time_request message,

* that the transmit_timestamp in that time_request message matches the corresponding time stamp from the synchronization data received in the time_response, and

* that the appended MAC verifies the received synchronization data, version number and nonce.

If at least one of the first three checks fails (i.e. if the version number does not match, if the client has never used the nonce transmitted in the time_response message or if it has used the nonce with initial time synchronization data different from that in the response), then the client MUST ignore this time_response message. If the MAC is invalid, the client MUST do one of the following: abort the run or go back to step 5 (because the cookie might have changed due to a server seed refresh). If both checks are successful, the client SHOULD continue time synchronization by going back to step 7.

The client's behavior in unicast mode is also expressed in Figure 1.

#### 7.1.2. The Client in Broadcast Mode

To establish a secure broadcast association with a broadcast server, the client MUST initially authenticate the broadcast server and securely synchronize its time to it up to an upper bound for its time offset in unicast mode. After that, the client performs the following steps:

1. It sends a client_bpar message to the server. It MUST remember the transmitted values for version number and signature algorithm.

2. It waits for a reply in the form of a server_bpar message after which it performs the following checks:

   * The message must contain all the necessary information for the TESLA protocol, as listed in Section 6.4.2.

   * Verification of the message's signature.

   If any information is missing or the server's signature cannot be verified, the client MUST abort the broadcast run. If all checks are successful, the client MUST remember all the broadcast parameters received for later checks.

3. The client awaits time synchronization data in the form of a server_broadcast message. Upon receipt, it performs the following checks:

    1. Proof that the MAC is based on a key that is not yet disclosed (packet timeliness). This is achieved via a combination of checks. First the disclosure schedule is used, which requires the loose time synchronization. If this is successful, the client gets a stronger guarantee via a key check exchange: it sends a client_keycheck message and waits for the appropriate response. Note that it needs to memorize the nonce and the time interval number that it sends as a correlated pair. For more detail on both of the mentioned timeliness checks, see Appendix Appendix C.4. If its timeliness is verified, the packet will be buffered for later authentication. Otherwise, the client MUST discard it. Note that the time information included in the packet will not be used for synchronization until its authenticity could also be verified.

    2. The client checks that it does not already know the disclosed key. Otherwise, the client SHOULD discard the packet to avoid a buffer overrun. If verified, the client ensures that the disclosed key belongs to the one-way key chain by applying the one-way function until equality with a previous disclosed key is shown. If falsified, the client MUST discard the packet.

    3. If the disclosed key is legitimate, then the client verifies the authenticity of any packet that it received during the corresponding time interval. If authenticity of a packet is verified it is released from the buffer and the packet's time information can be utilized. If the verification fails, then authenticity is no longer given. In this case the client MUST request authentic time from the server by means of a unicast time request message.

    See RFC 4082[RFC4082] for a detailed description of the packet verification process.

The client MUST restart the broadcast sequence with a client_bpar message Section 6.4.1 if the one-way key chain expires.

The client's behavior in broadcast mode can also be seen in Figure 2.

## 7.2. The Server

### 7.2.1. The Server in Unicast Mode

To support unicast mode, the server MUST be ready to perform the following actions:

- Upon receipt of a client_assoc message, the server constructs and sends a reply in the form of a server_assoc message as described in Section 6.1.2.

- Upon receipt of a client_cook message, the server checks whether it supports the given cryptographic algorithms. It then calculates the cookie according to the formula given in Section 5.1. With this, it MUST construct a server_cook message as described in Section 6.2.2.

- Upon receipt of a time_request message, the server re-calculates the cookie, then computes the necessary time synchronization data and constructs a time_response message as given in Section 6.3.2.

The server MUST refresh its server seed periodically (see Section 8.1).

### 7.2.2. The Server in Broadcast Mode

A broadcast server MUST also support unicast mode, in order to provide the initial time synchronization which is a precondition for any broadcast association. To support NTS broadcast, the server MUST additionally be ready to perform the following actions:

- Upon receipt of a client_bpar message, the server constructs and sends a server_bpar message as described in Section 6.4.2.

- Upon receipt of a client_keycheck message, the server looks up if it has already disclosed the key associated with the interval number transmitted in that message. If it has not disclosed it, it constructs and sends the appropriate server_keycheck message as described in Section 6.6.2. For more detail, see also Appendix C.

- The server follows the TESLA protocol in all other aspects, by regularly sending server_broad messages as described in Section 6.5.1, adhering to its own disclosure schedule.

It is also the server’s responsibility to watch for the expiration date of the one-way key chain and generate a new key chain accordingly.

# 8. Server Seed Considerations

The server has to calculate a random seed which has to be kept secret. The server MUST generate a seed for each supported hash algorithm, see Section 9.1.

## 8.1. Server Seed Refresh

According to the requirements in [RFC7384] the server MUST refresh each server seed periodically. As a consequence, the cookie memorized by the client becomes obsolete. In this case the client cannot verify the MAC attached to subsequent time response messages and has to respond accordingly by re-initiating the protocol with a cookie request (Section 6.2).

## 8.2. Server Seed Algorithm

## 8.3. Server Seed Lifetime

# 9. Hash Algorithms and MAC Generation

## 9.1. Hash Algorithms

Hash algorithms are used at different points: calculation of the cookie and the MAC, and hashing of the client's certificate. Client and server negotiate a hash algorithm H during the association message exchange (Section 6.1) at the beginning of a unicast run. The selected algorithm H is used for all hashing processes in that run.

In broadcast mode, hash algorithms are used as pseudo random functions to construct the one-way key chain. Here, the utilized hash algorithm is communicated by the server and non-negotiable.

The list of the hash algorithms supported by the server has to fulfill the following requirements:

- it MUST NOT include SHA-1 or weaker algorithms,
- it MUST include SHA-256 or stronger algorithms.

Note

Any hash algorithm is prone to be compromised in the future. A successful attack on a hash algorithm would enable any NTS client to derive the server seed from their own cookie. Therefore, the server MUST have separate seed values for its different supported hash algorithms. This way, knowledge gained from an attack on a

hash algorithm H can at least only be used to compromise such clients who use hash algorithm H as well.

### 9.2. MAC Calculation

For the calculation of the MAC, client and server are using a Keyed-Hash Message Authentication Code (HMAC) approach [RFC2104]. The HMAC is generated with the hash algorithm specified by the client (see Section 9.1).

## 10. IANA Considerations

## 11. Security Considerations

### 11.1. Initial Verification of the Server Certificates

The client has to verify the validity of the certificates during the certification message exchange (Section 6.1.2). Since it generally has no reliable time during this initial communication phase, it is impossible to verify the period of validity of the certificates. Therefore, the client MUST use one of the following approaches:

- The validity of the certificates is preconditioned. Usually this will be the case in corporate networks.
- The client ensures that the certificates are not revoked. To this end, the client uses the Online Certificate Status Protocol (OCSP) defined in [RFC6277].
- The client requests a different service to get an initial time stamp in order to be able to verify the certificates' periods of validity. To this end, it can, e.g., use a secure shell connection to a reliable host. Another alternative is to request a time stamp from a Time Stamping Authority (TSA) by means of the Time-Stamp Protocol (TSP) defined in [RFC3161].

### 11.2. Revocation of Server Certificates

According to Section 8.1, it is the client's responsibility to initiate a new association with the server after the server's certificate expires. To this end the client reads the expiration date of the certificate during the certificate message exchange (Section 6.1.2). Besides, certificates may also be revoked prior to the normal expiration date. To increase security the client MAY verify the state of the server's certificate via OCSP periodically.

### 11.3. Usage of NTP Pools

The certification based authentication scheme described in Section 6 is not applicable to the concept of NTP pools. Therefore, NTS is not able to provide secure usage of NTP pools.

### 11.4. Denial-of-Service in Broadcast Mode

TESLA authentication buffers packets for delayed authentication. This makes the protocol vulnerable to flooding attacks, causing the client to buffer excessive numbers of packets. To add stronger DoS protection to the protocol, client and server use the "not re-using keys" scheme of TESLA as pointed out in section 3.7.2 of RFC 4082 [RFC4082]. In this scheme the server never uses a key for the MAC generation more than once. Therefore the client can discard any packet that contains a disclosed key it knows already, thus preventing memory flooding attacks.

Note that an alternative approach to enhance TESLA’s resistance against DoS attacks involves the addition of a group MAC to each packet. This requires the exchange of an additional shared key common to the whole group. This adds additional complexity to the protocol and hence is currently not considered in this document.

### 11.5. Delay Attack

In a packet delay attack, an adversary with the ability to act as a MITM delays time synchronization packets between client and server asymmetrically [RFC7384]. This prevents the client to measure the network delay, and hence its time offset to the server, accurately [Mizrahi]. The delay attack does not modify the content of the exchanged synchronization packets. Therefore cryptographic means do not provide a feasible way to mitigate this attack. However, several non-cryptographic precautions can be taken in order to detect this attack.

1. Usage of multiple time servers: this enables the client to detect the attack provided that the adversary is unable to delay the synchronizations packets between the majority of servers. This approach is commonly used in NTP to exclude incorrect time servers [RFC5905].

2. Multiple communication paths: The client and server are utilizing different paths for packet exchange as described in the I-D [I-D.shpiner-multi-path-synchronization]. The client can detect the attack provided that the adversary is unable to manipulate the majority of the available paths [Shpiner]. Note that this approach is not yet available, neither for NTP nor for PTP.

3. Usage of an encrypted connection: the client exchanges all packets with the time server over an encrypted connection (e.g. IPsec). This measure does not mitigate the delay attack but it makes it more difficult for the adversary to identify the time synchronization packets.

4. For the unicast mode: Introduction of a threshold value for the delay time of the synchronization packets. The client can discard a time server if the packet delay time of this time server is larger than the threshold value.

Additional provision against delay attacks has to be taken in the broadcast mode. This mode relies on the TESLA scheme which is based on the requirement that a client and the broadcast server are loosely time synchronized. Therefore, a broadcast client has to establish time synchronization with its broadcast server before it maintains time synchronization by utilization of the broadcast mode. To this end it initially establishes a unicast association with its broadcast server until time synchronization and calibration of the packet delay time is achieved. After that it establishes a broadcast association to the broadcast server and utilizes TESLA to verify integrity and authenticity of any received broadcast packets.

An adversary who is able to delay broadcast packets can cause a time adjustment at the receiving broadcast clients. If the adversary delays broadcast packets continuously, then the time adjustment will accumulate until the loose time synchronization requirement is violated, which breaks the TESLA scheme. To mitigate this vulnerability the security condition in TESLA has to be supplemented by an additional check in which the client, upon receipt of a broadcast message, verifies the status of the corresponding key via a unicast message exchange with the broadcast server (see section Appendix C.4 for a detailed description of this check). Note, that a broadcast client should also apply the above mentioned precautions as far as possible.

## 12. Acknowledgements

The authors would like to thank Russ Housley, Steven Bellovin, David Mills and Kurt Roeckx for discussions and comments on the design of NTS. Also, thanks to Harlan Stenn for his technical review and specific text contributions to this document.

## 13. References

### 13.1. Normative References

[IEEE1588] IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems", 2008.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC6277] Santesson, S. and P. Hallam-Baker, "Online Certificate Status Protocol Algorithm Agility", RFC 6277, June 2011.

### 13.2. Informative References

[I-D.shpiner-multi-path-synchronization] Shpiner, A., Tse, R., Schelp, C., and T. Mizrahi, "Multi-Path Time Synchronization", draft-shpiner-multi-path-synchronization-03 (work in progress), February 2014.

[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, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, October 2014.

[Shpiner] Shpiner, A., Revah, Y., and T. Mizrahi, "Multi-path Time Protocols", in Proceedings of Precision Clock Synchronization for Measurement Control and Communication, ISPCS 2013, pp. 1-6, September 2013.

## Appendix A. Flow Diagrams of Client Behaviour

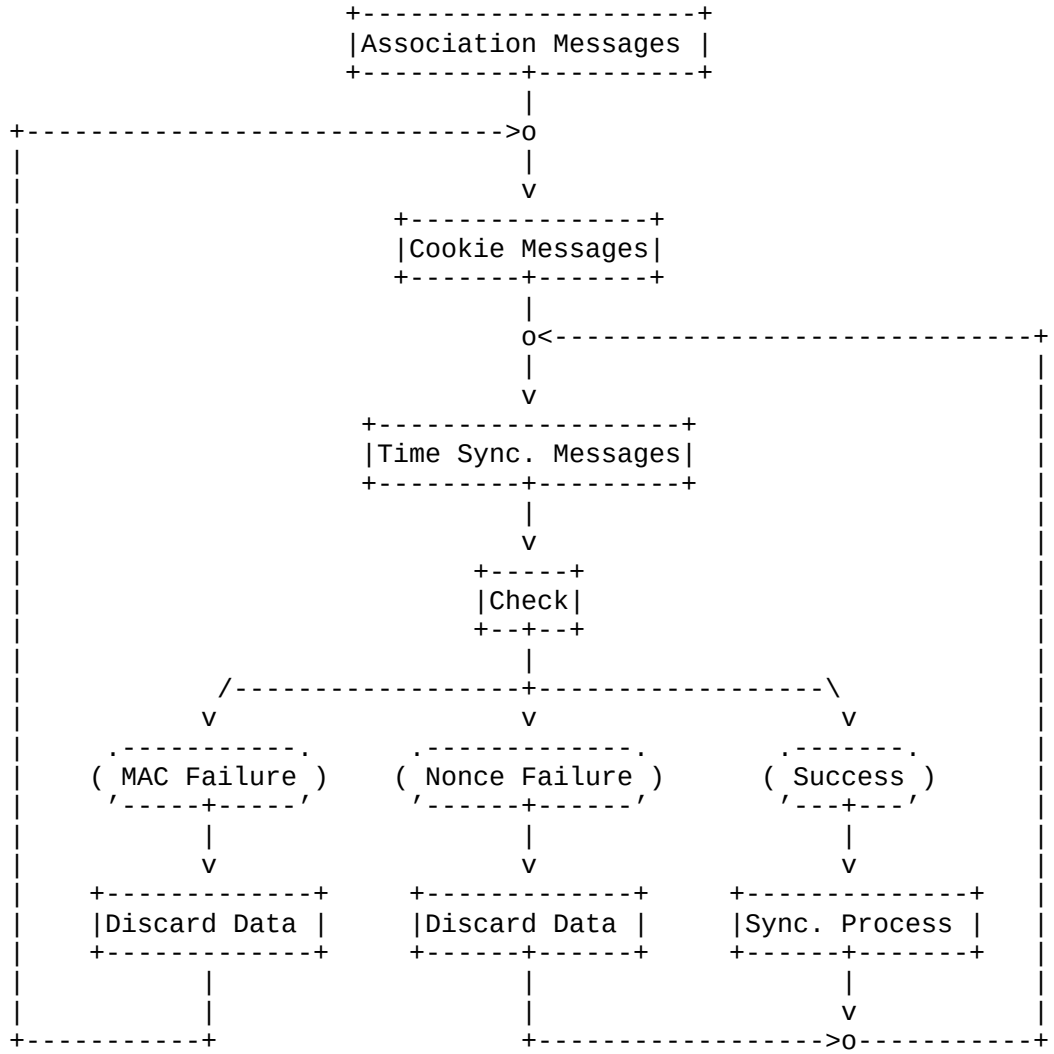



Figure 1: The client's behavior in NTS unicast mode.

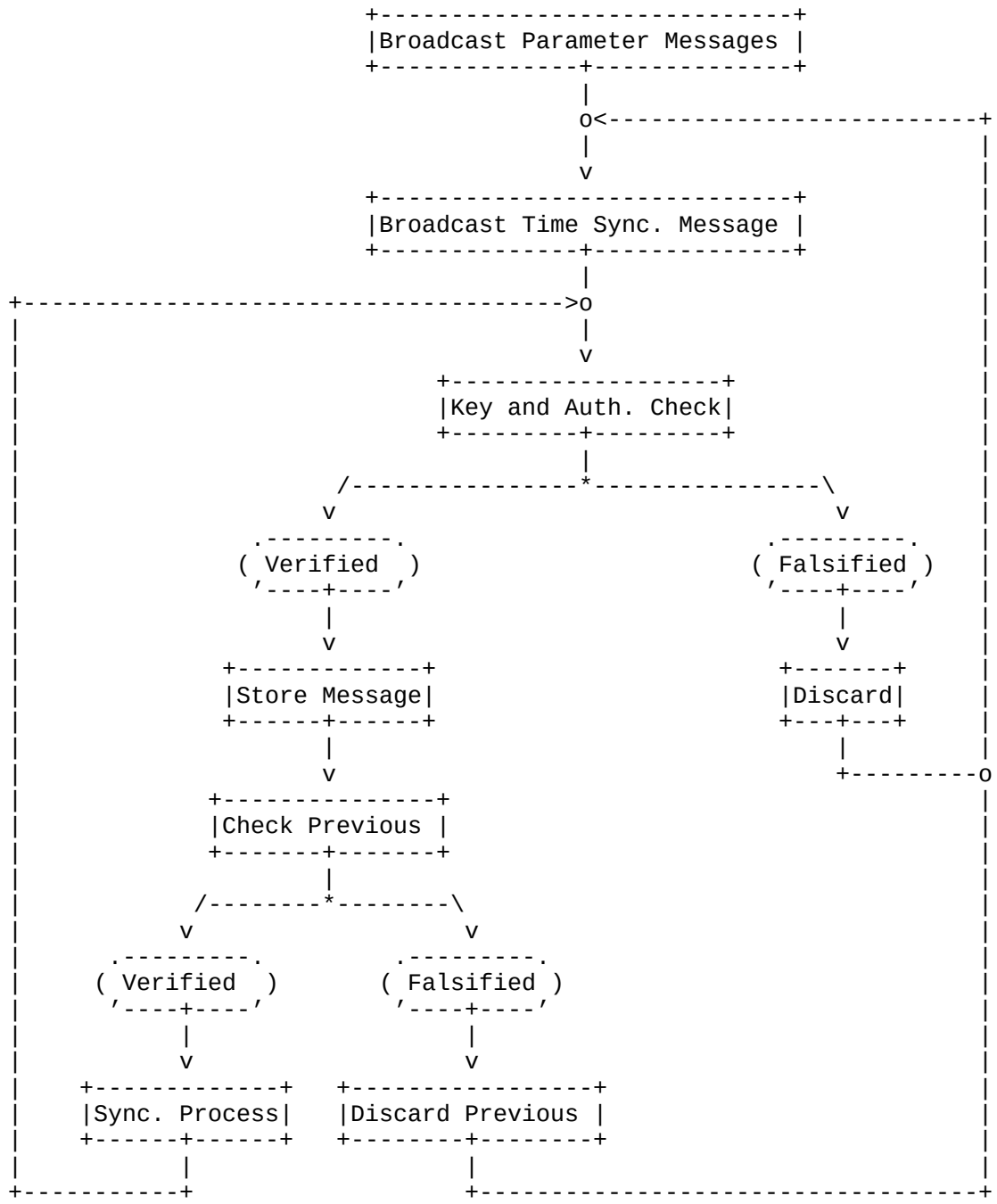



Figure 2: The client’s behaviour in NTS broadcast mode.

## Appendix B. TICTOC Security Requirements

The following table compares the NTS specifications against the TICTOC security requirements [RFC7384].

| Section | Requirement from I-D tictoc security-requirements-05 | Requirement level | NTS |
|---|---|---|---|
| 5.1.1 | Authentication of Servers | MUST | OK |
| 5.1.1 | Authorization of Servers | MUST | OK |
| 5.1.2 | Recursive Authentication of Servers (Stratum 1) | MUST | OK |
| 5.1.2 | Recursive Authorization of Servers (Stratum 1) | MUST | OK |
| 5.1.3 | Authentication and Authorization of Slaves | MAY | - |
| 5.2 | Integrity protection. | MUST | OK |
| 5.4 | Protection against DoS attacks | SHOULD | OK |
| 5.5 | Replay protection | MUST | OK |
| 5.6 | Key freshness. | MUST | OK |
| | Security association. | SHOULD | OK |
| | Unicast and multicast associations. | SHOULD | OK |
| 5.7 | Performance: no degradation in quality of time transfer. | MUST | OK |
| | Performance: lightweight computation | SHOULD | OK |
| | Performance: storage, bandwidth | SHOULD | OK |
| 5.7 | Confidentiality protection | MAY | NO |
| 5.9 | Protection against Packet Delay and Interception Attacks | SHOULD | NA*) |

| 5.10 | Secure mode | MUST | - |
|---|---|---|---|
| | Hybrid mode | SHOULD | - |

*) See discussion in section Section 11.5.

Comparison of NTS sepecification against TICTOC security requirements.

## Appendix C. Broadcast Mode

For the broadcast mode, NTS adopts the TESLA protocol with some customizations. This appendix provides details on the generation and usage of the one-way key chain collected and assembled from [RFC4082]. Note that NTS is using the "not re-using keys" scheme of TESLA as described in section 3.7.2. of [RFC4082].

### C.1. Server Preparations

Server setup:

1. The server determines a reasonable upper bound B on the network delay between itself and an arbitrary client, measured in milliseconds.

2. It determines the number n+1 of keys in the one-way key chain. This yields the number n of keys that are usable to authenticate broadcast packets. This number n is therefore also the number of time intervals during which the server can send authenticated broadcast messages before it has to calculate a new key chain.

3. It divides time into n uniform intervals I_1, I_2, ..., I_n. Each of these time intervals has length L, measured in milliseconds. In order to fulfill the requirement 3.7.2. of RFC 4082 the time interval L has to be smaller than the time interval between the broadcast messages.

4. The server generates a random key K_n.

5. Using a one-way function F, the server generates a one-way chain of n+1 keys K_0, K_1, ..., K_{n} according to

   K_i = F(K_{i+1}).

6. Using another one-way function F', it generates a sequence of n+1 MAC keys K'_0, K'_1, ..., K'_{n-1} according to

K’_i = F’(K_i).

7. Each MAC key K’_i is assigned to the time interval I_i.

8. The server determines the key disclosure delay d, which is the number of intervals between using a key and disclosing it. Note that although security is provided for all choices d>0, the choice still makes a difference:

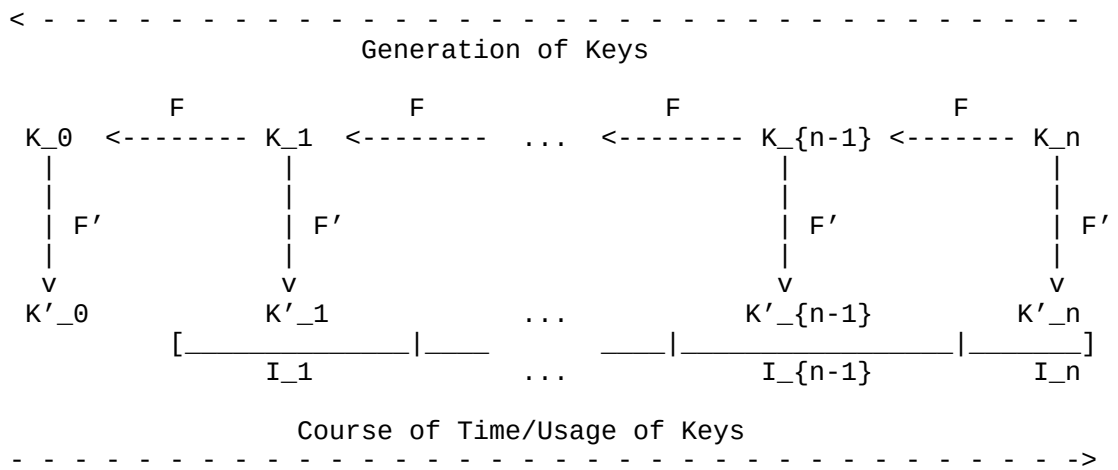
   * If d is chosen too short, the client might discard packets because it fails to verify that the key used for their MAC has not been yet disclosed.

   * If d is chosen too long, the received packets have to be buffered for a unnecessarily long time before they can be verified by the client and subsequently be utilized for time synchronization.

   The server SHOULD calculate d according to

      d = ceil( 2*B / L) + 1,

   where ceil gives the smallest integer greater than or equal to its argument.




A Schematic explanation on the TESLA protocol’s one-way key chain

### C.2. Client Preparation

A client needs the following information in order to participate in a TESLA broadcast.

- One key K_i from the one-way key chain, which has to be authenticated as belonging to the server. Typically, this will be K_0.
- The disclosure schedule of the keys. This consists of:
  - the length n of the one-way key chain,
  - the length L of the time intervals I_1, I_2, ..., I_n,
  - the starting time T_i of an interval I_i. Typically this is the starting time T_1 of the first interval;
  - the disclosure delay d.
- The one-way function F used to recursively derive the keys in the one-way key chain,
- The second one-way function F' used to derive the MAC keys K'_0, K'_1, ... , K'_n from the keys in the one-way chain.
- An upper bound D_t on how far its own clock is "behind" that of the server.

Note that if D_t is greater than (d - 1) * L, then some authentic packets might be discarded. If D_t is greater than d * L, then all authentic packets will be discarded. In the latter case, the client should not participate in the broadcast, since there will be no benefit in doing so.

### C.3. Sending Authenticated Broadcast Packets

During each time interval I_i, the server sends one authenticated broadcast packet P_i. This packet consists of:

- a message M_i,
- the index i (in case a packet arrives late),
- a MAC authenticating the message M_i, with K'_i used as key,
- the key K_{i-d}, which is included for disclosure.

### C.4. Authentication of Received Packets

When a client receives a packet P_i as described above, it first checks that it has not received a packet with the same disclosed key before. This is done to avoid replay/flooding attacks. A packet that fails this test is discarded.

Next, the client begins to check the packet's timeliness by ensuring that, according to the disclosure schedule and with respect to the upper bound D_t determined above, the server cannot have disclosed the key K_i yet. Specifically, it needs to check that the server's clock cannot read a time that is in time interval I_{i+d} or later. Since it works under the assumption that the server's clock is not more than D_t "ahead" of the client's clock, the client can calculate an upper bound t_i for the server's clock at the time when P_i arrived. This upper bound t_i is calculated according to

$$t_i = R + D_t,$$

where R is the client's clock at the arrival of P_i. This implies that at the time of arrival of P_i, the server could have been in interval I_x at most, with

$$x = \mathrm{floor}((t_i - T_1) / L) + 1,$$

where floor gives the greatest integer less than or equal to its argument. The client now needs to verify that

$$x < i+d$$

is valid (see also section 3.5 of [RFC4082]). If falsified, it is discarded.

If the check above is successful, the client performs another more rigorous check: it sends a key check request to the server (in the form of a client_keycheck message), asking explicitly if K_i has already been disclosed. It remembers the timestamp t_check of the sending time of that request as well as the nonce it used correlated with the interval number i. If it receives an answer from the server stating that K_i has not yet been disclosed and it is able to verify the HMAC on that response, then it deduces that K_i was undisclosed at t_check and therefore also at R. In this case, the clients accepts P_i as timely.

Next the client verifies that a newly disclosed key K_{i-d} belongs to the one-way key chain. To this end it applies the one-way function F to K_{i-d} until it can verify identity with an earlier disclosed key (see Clause 3.5 in RFC 4082, item 3).

Next the client verifies that the transmitted time value s_i belongs to the time interval I_i, by checking

T_i =< s_i, and

s_i < T_{i+1}.

If falsified, the packet MUST be discarded and the client MUST reinitialize the broadcast mode with a unicast association (because a falsification of this check yields that the packet was not generated according to protocol, which suggests an attack).

If a packet P_i passes all tests listed above, it is stored for later authentication. Also, if at this time there is a package with index i-d already buffered, then the client uses the disclosed key K_{i-d} to derive K'_{i-d} and uses that to check the MAC included in package P_{i-d}. On success, it regards M_{i-d} as authenticated.

## Appendix D. Random Number Generation

At various points of the protocol, the generation of random numbers is required. The employed methods of generation need to be cryptographically secure. See [RFC4086] for guidelines concerning this topic.

## Authors' Addresses

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

Stephen Roettger
Google Inc

Email: stephen.roettger@googlemail.com

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

Phone: +49-(0)531-592-8421
Email: kristof.teichel@ptb.de

TICTOC Working Group
Internet-Draft
Intended status: Experimental
Expires: April 18, 2016

S. Davari
A. Oren
Broadcom Corp.
M. Bhatia
P. Roberts
Alcatel-Lucent
L. Montini
Cisco Systems

October 16, 2015

# Transporting Timing messages over MPLS Networks
## draft-ietf-tictoc-1588overmpls-07

## Abstract

This document defines a method for transporting timing messages, such as Precision Time Protocol (PTP) or Network Time Protocol (NTP), over a Multiprotocol Label Switched (MPLS) network. The method facilitates efficient recognition of timing packets to enable their port level processing in both Label Edge Routers (LERs) and Label Switched Routers (LSRs).

The basic mechanism is to transport timing messages inside "Timing LSPs", which are dedicated MPLS Label Switched Paths (LSPs) that carry only timing, and possibly related Operations, Administration and Maintenance (OAM) or management packets, but do not carry customer traffic.

Two encapsulations methods are defined. The first transports UDP/IP encapsulated timing messages directly over the dedicated LSP. The second transports Ethernet encapsuled timing messages inside an Ethernet pseudowire.

## Status of This Memo

This Internet-Draft will expire on April 18, 2016.



Table of Contents

## 1. Introduction

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 [RFC2119].

When used in lower case, these words convey their typical use in common language, and are not to be interpreted as described in RFC2119 [RFC2119].

The objective of timing distribution protocols, such as Precision Time Protocol (PTP) and Network Timing Protocol (NTP), is to synchronize clocks running on nodes of a distributed system.

Timing distribution protocols are presently transported over IP or Ethernet. The present document presents a mechanism for transport over Multiprotocol Label Switched (MPLS) networks. Our solution involves transporting timing messages over dedicated "Timing Label Switched Paths (LSPs)". These are ordinary LSPs that carry timing messages and MAY carry Operations, Administration and Maintenance (OAM) or management messages, but do not carry any other traffic.

Timing LSPs may be established statically or via signaling. When using signaling, extensions to routing protocols (e.g., OSPF, ISIS) are required to enable routers to distribute their timing processing capabilities, and extensions to path set up protocols (e.g., RSVP-TE) are required for establishing the LSPs. All such extensions are beyond the scope of this document.

High accuracy timing distribution requires on-path support, e.g., Transparent Clocks (TCs) or Boundary Clocks (BCs), at intermediate nodes. These intermediate nodes need to recognize and appropriately process timing distribution packets. To facilitate efficient recognition of timing messages transported over MPLS, this document restricts the specific encapsulations to be used.

[IEEE-1588] defines PTP messages for frequency, phase and time synchronization. PTP messages may be transported over UDP/IP (Annex D and E of [IEEE-1588]) or over Ethernet (Annex F of [IEEE-1588]). This document defines two methods to transport PTP messages over MPLS networks.

PTP defines several clock types, including ordinary clocks, boundary clocks, end-to-end transparent clocks, and peer-to-peer transparent clocks. Transparent clocks are situated at intermediate nodes and

update the Correction Field inside PTP messages in order to reflect the time required to transit the node.

[RFC5905] defines NTP messages for clock and time synchronization. NTP messages are transported over UDP/IP. This document defines a method to transport NTP messages over MPLS networks.

It can be expected that only a subset of LSR ports will be capable of processing timing messages. Timing LSPs MUST be set up (either by manual provisioing or via signaling) to traverse these ports. While Timing LSPs are designed to optimize timing distribution, the performance of slave clocks is beyond the scope of this document.

Presently on-path support is only defined for PTP, and therefore much of our discussion will focus on PTP. NTP timing distribution may benefit from transport in a Timing LSP due to prioritorization or selection of ports or nodes with minimal delay or delay asymmetry.

## 2. Terminology

1588: The timing distribution protocol defined in IEEE 1588.

Boundary Clock: A device with one timing port to receive timing messages and at least one port to re-distribute timing messages.

CF: Correction Field, a field inside certain PTP messages that holds the accumulated transit time.

Master Clock: The source of 1588 timing messages to a set of slave clocks.

NTP: The timing distribution protocol defined in RFC 5905.

Ordinary Clock: A master or slave clock. Note that ordinary clocks have only a single PTP port.

PTP: Precision Time Protocol. See 1588.

Slave Clock: A receiver of 1588 timing messages from a master clock.

Timing LSP: An MPLS LSP dedicated to carry timing messages.

Timing messages: Timing distribution protocol messages that are exchanged between clocks.

Timing port: A port on a (master, slave, transparent, or boundary) clock.

Timing PW: A PW within a Timing LSP that is dedicated to carry timing messages.

Transparent Clock: An intermediate node that forwards timing messages while updating their CF.

## 3. Problem Statement

[IEEE-1588] defines methods for transporting PTP messages over Ethernet and IP networks. [RFC5905] defines a method of transporting NTP messages over IP networks. There is a need to transport timing messages over MPLS networks while supporting the Transparent Clock (TC), Boundary Clock (BC) and Ordinary Clock (OC) functionalities in LER and LSRs of the MPLS network.

There are potentially many ways of transporting timing packets over MPLS. However, it is advisable to limit the number of possible encapsulation options to simplify recognition and processing of timing packets.

The solution herein desscribed transports timing messages over dedicated "Timing Label Switched Paths (LSPs)". Were timing packets to share LSPs with other traffic, intermediate LSRs would be required to perform some deeper inspection to differentiate between timing packets and other packets. The method herein proposed avoids this complexity, and can readily detect all PTP messages (one-step or two-step), and supports ordinary, boundary and transparent clocks.

## 4. Timing over MPLS Architecture

Timing messages are exchanged between timing ports on ordinary and boundary clocks. Boundary clocks terminate the timing messages and act as master clock for other boundary clocks or slave clocks. End-to-End transparent clocks do not terminate the timing messages but do modify the contents of the timing messages in transit.

OC, BC and TC functionality may be implemented in either LERs or LSRs.

An example is shown in Figure 1, where the LERs act as OCs and are the initiating/terminating points for timing messages. The ingress LER encapsulates timing messages in a Timing LSP and the egress LER terminates this Timing LSP. Intermediate LSRs (only one is shown here) act as TCs, updating the CF of transiting timing messages, as well as performing label switching operations.

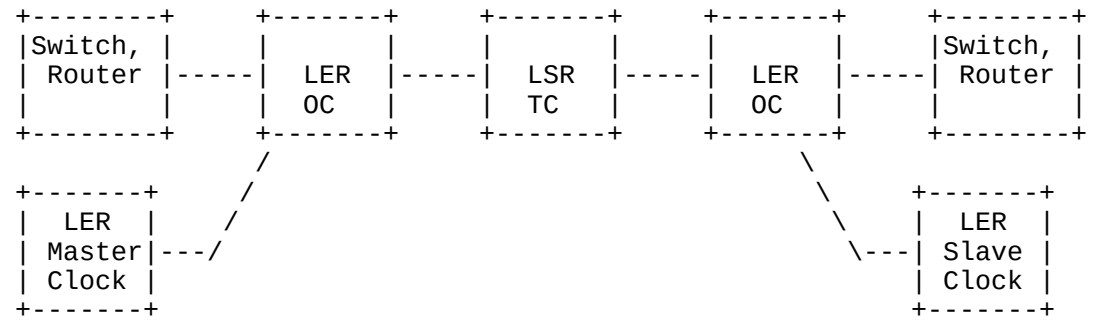



Figure (1) - Deployment example 1 of timing over MPLS network

Another example is shown in Figure 2, where LERs act as BCs, and switches/routers outside of the MPLS network, act as OCs or BCs. The ingress LER BC recovers timing and initiates timing messages encapsulated in the Timing LSP toward the MPLS network, an intermediate LSR acts as a TC, and the egress LER acts as a BC sending timing messages to equipment outside the MPLS network.

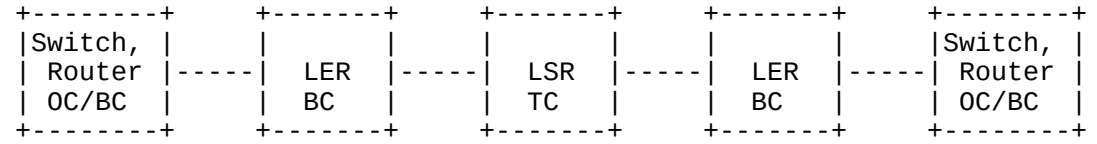



Figure (2) - Deployment example 2 of timing over MPLS network

Yet another example is shown in Figure 3, where both LERs and LSRs act as TCs. The ingress LER updates the CF and encapsulates the timing message in an MPLS packet, intermediate LSRs update the CF and perform label switching, and the egress LER updates the CF and sends the timing messages to equipment outside the MPLS network.

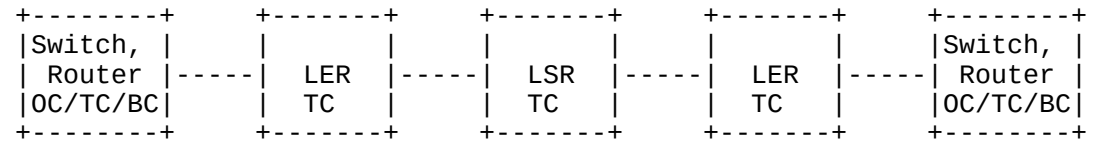



Figure (3) - Deployment example 3 of timing over MPLS network

A final example is shown in Figure 4, where all nodes act as BCs. Single-hop LSPs are created between every two adjacent LSRs. Of course, PTP transport over Ethernet MAY be used between two network elements.

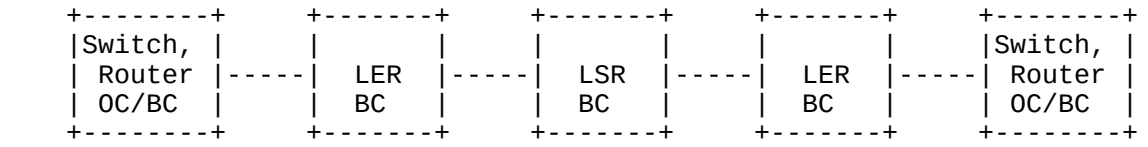



Figure (4) - Deployment example 3 of timing over MPLS network

An MPLS domain MAY serve multiple customers, each having its own Timing domain. In these cases the MPLS domain (maintained by a service provider) MUST provide dedicated timing services to each customer.

The timing over MPLS architecture assumes a full mesh of Timing LSPs between all LERs supporting this specification. It supports point-to- point (VPWS) and Multipoint (VPLS) services. This means that a customer may purchase a point-to-point timing service between two customer sites or a multipoint timing service between more than two customer sites.

The Timing over MPLS architecture supports P2P or P2MP Timing LSPs. This means that the Timing Multicast messages such as PTP Multicast event messages MAY be transported over P2MP Timing LSPs or MAY be replicated and transported over multiple P2P Timing LSPs.

Timing LSPs, as defined by this specification, MAY be used for timing messages that do not require time-stamping or CF updating.

PTP Announce messages that determine the Timing LSP terminating point behavior such as BC/OC/TC SHOULD be transported over the Timing LSP to simplify hardware and software.

## 5. Dedicated LSPs for Timing messages

The method defined in this document is used by LER and LSRs to identify timing messages by observing the top label of the MPLS label stack. Compliant implementations MUST use dedicated LSPs to carry timing messages over MPLS. Such LSPs are herein referred to as "Timing LSPs" and the labels associated with these LSPs as "Timing LSP labels".

Timing distribution requires symmetrical bidirectional communications. Co-routing of the two directions is required to limit delay asymmetry. Thus timing messages MUST be transported either over two co-routed unidirectional Timing LSPs, or a single bidirectional co-routed Timing LSP.

Timing LSPs MAY be configured using RSVP-TE. Extensions to RSVP-TE are required for this purpose, but are beyond the scope of this document.

## 6. Timing over LSP Encapsulation

We define two methods for carrying timing messages over MPLS. The first method transports UDP/IP-encapsulated timing messages over Timing LSPs, and the second method transports Ethernet encapsulated timing messages over Ethernet PWs placed in Timing LSPs.

### 6.1. Timing over UDP/IP over MPLS Encapsulation

The first method directly encapsulates UDP/IP timing messages in a Timing LSP. The UDP/IP encapsulation of PTP messages MUST comply to Annex D and E of [IEEE-1588], and the UDP/IP encapsulation of NTP messages MUST comply to [RFC5905]. This format is shown in Figure 4.

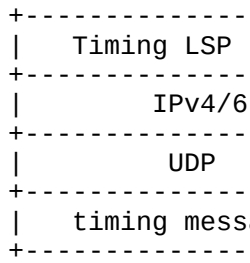



Figure (4) - Timing over UDP/IP over MPLS Encapsulation

In order for an LER/LSR to process timing messages, the Timing LSP Label must be the top label of the label stack. The LER/LSR MUST know that this label is a Timing LSP Label. It can learn this by static configuration or via RSVP-TE signaling.

### 6.2. Timing over PW Encapsulation

Another method of transporting timing over MPLS networks is to use Ethernet encapsulated timing messages, and to transport these in an Ethernet PW which in turn is transported over a Timing LSP. In the

case of PTP, the Ethernet encapsulation MUST comply to Annex F of [IEEE-1588] and the Ethernet PW encapsulation to [RFC4448], resulting in the format shown in Figure 5(A).

Either the Raw mode or Tagged mode defined in [RFC-4448] MAY be used and the payload MAY have 0, 1, or 2 VLAN tags. The Timing over PW encapsulation MUST use the Control Word (CW) as specified in [RFC4448]. The use of Sequence Number in the CW is optional.

NTP MAY be transported using an IP PW (as defined in [RFC4447]) as shown in Fig 5(B).

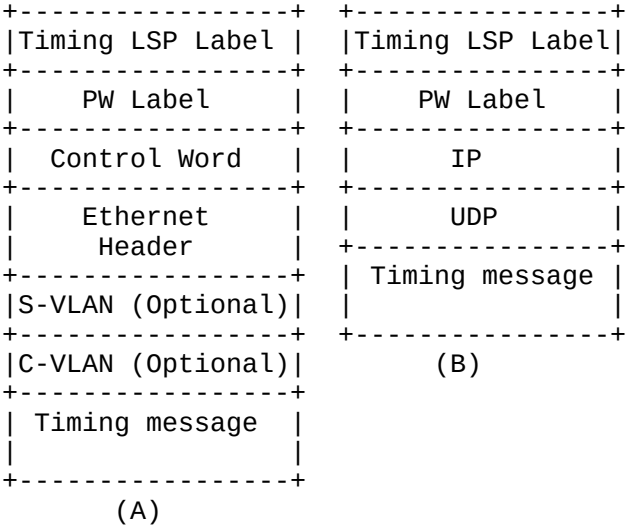



Figure (5) - Timing over PW Encapsulations

## 7. Timing message Processing

Each Timing protocol such as PTP and NTP, defines a set of timing messages. PTP defines SYNC, DELAY_REQ, DELAY_RESP, FOLLOW_UP, etc.

Some timing messages require per-packet processing, such as time-stamping or CF updating. A compliant LER/LSR parses each timing message to determine the required processing.

For example, the following PTP messages (event messages) require time-stamping or CF updating:

o SYNC

- DELAY_REQ (Delay Request)
- PDELAY_REQ (Peer Delay Request)
- PDELAY_RESP (Peer Delay Response)

SYNC and DELAY_REQ are exchanged between a Master Clock and a Slave Clock and MUST be transported over Timing LSPs. PDELAY_REQ and PDELAY_RESP are exchanged between adjacent PTP clocks (master, slave, boundary, or transparent) and SHOULD be transported over single hop Timing LSPs. If two-Step PTP clocks are present, then the FOLLOW_UP, and PDELAY_RESP_FOLLOW_UP messages MUST also be transported over Timing LSPs.

For a given instance of the 1588 protocol, SYNC and DELAY_REQ MUST be transported in opposite directions. As aforementioned, two co-routed unidirectional LSPs or a single bidirectional co-routed LSP MAY be used.

Except as indicated above for two-step PTP clocks, PTP messages that are not "event messages" need not be processed by intermediate routers. These message types MAY be carried in PTP Tunnel LSPs.

## 8. Protection and Redundancy

In order to ensure continuous uninterrupted operation of timing distribution, slave clocks often track redundant master clocks. Prolonged outages of Timing LSPs trigger switching to a redundant master clock It is the responsibility of the network operator to ensure that physically disjoint Timing LSPs are established between a slave clock and redundant master clocks.

LSP or PW layer protection, such as linear protection Switching, ring protection switching or MPLS Fast Reroute (FRR), will lead to changes in propagation delay between master and slave clocks. Such a change, if undetected by the slave clock, would negatively impact timing performance. While it is expected that slave clocks will often be able to detect such delay changes, this specification RECOMMENDS that automatic protection switching NOT be used for Timing LSPs, unless the operator can ensure that it will not negatively impact timing performance.

## 9. ECMP and Entropy

To ensure the correct operation of slave clocks and avoid error introduced by forward and reverse path delay asymmetry, the physical path taken by timing messages MUST be the same for all timing

messages. In particular, the PTP event messages listed in section 7 MUST be routed in the same way.

Therefore the Timing LSPs MUST not be subject to ECMP (Equal Cost Multipath). Entropy labels MUST NOT be used for the Timing LSP [RFC6790] and MUST NOT be used for PWs inside the Timing LSP [RFC6391].

## 10. PHP

To ensure that the label on the top of the label stack is the Timing LSP Label, PHP MUST not be employed.

## 11. OAM, Control and Management

In order to monitor Timing LSPs or PWs, it is necessary to enable them to carry OAM messages. OAM packets MUST be differentiated from timing messages by already defined IETF methods.

For example BFD [RFC5880], [RFC5884] and LSP-Ping [RFC4389] MAY run over Timing LSPs via UDP/IP encapsulation or via GAL/G-ACh. These protocols can easily be identified by the UDP Destination port number or by GAL/G-ACh respectively.

Also BFD, LSP-Ping and other messages MAY run over Timing PWs via VCCV [RFC5085]. In this case these messages are recognized according to the VCCV type.

## 12. QoS Considerations

There may be deployments where timing messages traverse LSR/LERs that are not capable of the required processing. In order to minimize the negative impact on the timing performance of the slave clock timing messages MUST be treated with the highest priority. This can be achieved by proper setup of Timing LSPs.

It is recommended that Timing LSPs be configured to indicate EF-PHB [RFC3246] for the CoS and "green" [RFC2697] for drop eligibility.

## 13. FCS and Checksum Recalculation

Since Boundary and Transparent Clocks modify packets, when the MPLS packets are transported over Ethernet the processing MUST include recalculation of the Ethernet FCS. FCS retention as described in [RFC4720] MUST NOT be used.

For the UDP/IP encapsulation mode, calculation of the UDP checksum will generally be required. After updating the CF a Transparent

Clock MUST either incrementally update the UDP checksum or completely recalculate the checksum before transmission to downstream node.

## 14. Behavior of LER/LSRs

Timing-aware LERs or LSRs are MPLS routers that are able to recognize timing packets. Timing-capable LERs and LSRs further have one or more interfaces that can perform timing processing (OC/BC/TC) on timing packets. Timing-capable/aware LERs and LSRs MAY advertise the timing capabilities of their interfaces via control plane protocols such as OSPF or IS-IS, and timing-aware LERs can then be set up Timing LSPs via RSVP-TE signaling. Alternatively the timing capabilities of LERs and LSRs may be known by a centralized controller or management system, and Timing LSPs may be manually configured, or set up by a management platform or a Software Defined Networking (SDN) controller.

### 14.1. Behavior of Timing-capable/aware LERs/LSRs

When a timing-capable ingress LER acting as a TC receives a timing message packet from a timing-capable non-MPLS interface, the LER updates the CF, encapsulates and forwards the packet over a previously established Timing LSP. When a timing-capable egress LER acting as a TC receives a timing message packet on timing-capable MPLS interface, the LER updates the CF, decapsulates the MPLS encapsulation, and forwards the packet via a non-MPLS interface. When a timing-capable LSR acting as a TC receives a timing message from a timing-capable MPLS interface, the LSR updates the CF and forwards the timing message over another MPLS interface.

When a timing-capable LER acting as a BC receives a timing message packet from a timing-capable interface, the LER time-stamps the packet and sends it to the BC processing module.

When a timing-capable LER acting as an OC receives a timing message from a timing-capable MPLS interface, the LER time-stamps the packet and sends it to the OC processing module.

### 14.2. Behavior of non-Timing-capable/aware LSR

It is most beneficial when all LSRs in the path of a Timing LSP be timing-Capable/aware LSRs. This would ensure the highest quality time and clock synchronization by slave clocks. However, this specification does not mandate that all LSRs in path of a Timing LSP be timing-capable/aware.

Non-timing-capable/aware LSRs just perform label switching on the packets encapsulated in Timing LSPs and don’t perform any timing

related processing. However, as explained in QoS section, timing packets MUST be still be treated with the highest priority based on their Traffic Class marking.

## 15. Other considerations

[IEEE-1588] defines an optional peer-to-peer transparent clocking (P2P TC) mode that compensates both for residence time in the network node and for propagation time on the link between modes. To support P2P TC, delay measurement must be performed between two adjacent timing-capable/aware LSRs. Thus, in addition to the TC functionality detailed above on transit PTP timing messages, adjacent peer to peer TCs MUST engage in single-hop peer delay measurement.

For single hop peer delay measurement a single-hop LSP SHOULD be created between the two adjacent LSRs. Other methods MAY be used; for example, if the link between the two adjacent routers is Ethernet, PTP transport over Ethernet MAY be used.

To support P2P TC, a timing-capable/ware LSR MUST maintain a list of all neighbors to which it needs to send a PDelay_Req, and maintain a single-hop timing LSP to each.

The use of Explicit Null Label (label 0 or 2) is acceptable as long as either the Explicit Null label is the bottom of stack label (for the UDP/IP encapsulation) or the label below the Explicit Null label (for the PW case).

## 16. Security Considerations

Security considerations for MPLS and pseudowires are discussed in [RFC3985] and [RFC4447]. Security considerations for timing are discussed in [RFC7384]. Everything discussed in those documents applies to the Timing LSP of this document.

An experimental security protocol is defined in [IEEE-1588]. The PTP security extension and protocol provides group source authentication, message integrity, and replay attack protection for PTP messages.

When the MPLS network (provider network) serves multiple customers, it is important to distinguish between timing messages belonging to different customers. For example if an LER BC is synchronized to a grandmaster belonging to customer A, then the LER MUST only use that BC for slaves of customer A, to ensure that customer A cannot adversely affect the timing distribution of other customers.

Timing messages MAY be encrypted or authenticated, provided that the timing-capable LERs/LSRs can authenticate/ decrypt the timing messages.

## 17. Applicability Statement

The Timing over MPLS transport methods described in this document apply to the following network Elements:

- An ingress LER that receives IP or Ethernet encapsulated timing messages from a non-MPLS interface and forwards them as MPLS encapsulated timing messages over Timing LSP, optionally performing TC functionality.
- An egress LER that receives MPLS encapsulated timing messages from a Timing LSP and forwards them to non-MPLS interface as IP or Ethernet encapsulated timing messages, optionally performing TC functionality.
- An ingress LER that receives MPLS encapsulated timing messages from a non-MPLS interface, performs BC functionality, and sends timing messages over a Timing LSP.
- An egress LER that receives MPLS encapsulated timing messages from a Timing LSP, performs BC functionality, and sends timing messages over a non-MPLS interface.
- An LSR on a Timing LSP that receives MPLS encapsulated timing messages from one MPLS interface and forwards them to another MPLS interface, optionally performing TC functionality.

This document also supports the case where not all LSRs are timing-capable/aware, or not all LER/LSR interfaces are timing-capable/ aware.

## 18. Acknowledgements

The authors would like to thank Yaakov Stein, Luca Martini, Ron Cohen, Tal Mizrahi, Stefano Ruffini, Peter Meyer and other IETF participants for reviewing and providing feedback on this draft.

## 19. IANA Considerations

There are no IANA requirements in this specification.

# 20. References

## 20.1. Normative References

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

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

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <http://www.rfc-editor.org/info/rfc3985>.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <http://www.rfc-editor.org/info/rfc4389>.

[RFC4447] Martini, L., Ed., Rosen, E., El-Aawar, N., Smith, T., and G. Heron, "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", RFC 4447, DOI 10.17487/RFC4447, April 2006, <http://www.rfc-editor.org/info/rfc4447>.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <http://www.rfc-editor.org/info/rfc4448>.

[RFC4720] Malis, A., Allan, D., and N. Del Regno, "Pseudowire Emulation Edge-to-Edge (PWE3) Frame Check Sequence Retention", RFC 4720, DOI 10.17487/RFC4720, November 2006, <http://www.rfc-editor.org/info/rfc4720>.

[RFC5085] Nadeau, T., Ed. and C. Pignataro, Ed., "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, DOI 10.17487/RFC5085, December 2007, <http://www.rfc-editor.org/info/rfc5085>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <http://www.rfc-editor.org/info/rfc5880>.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <http://www.rfc-editor.org/info/rfc5884>.

## 20.2. Informative References

[ISO] ISO/IEC 10589:1992, "Intermediate system to Intermediate system routing information exchange protocol for use in conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473)", April 1992.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <http://www.rfc-editor.org/info/rfc1195>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <http://www.rfc-editor.org/info/rfc2328>.

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <http://www.rfc-editor.org/info/rfc2697>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <http://www.rfc-editor.org/info/rfc3246>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <http://www.rfc-editor.org/info/rfc5340>.

[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, <http://www.rfc-editor.org/info/rfc5905>.

[RFC6391] Bryant, S., Ed., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, DOI 10.17487/RFC6391, November 2011, <http://www.rfc-editor.org/info/rfc6391>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <http://www.rfc-editor.org/info/rfc6790>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <http://www.rfc-editor.org/info/rfc7384>.

# Appendix A. Appendix

## A.1. Routing extensions for Timing-aware Routers

MPLS-TE routing relies on extensions to OSPF [RFC2328] [RFC5340] and IS-IS [ISO] [RFC1195] in order to advertise Traffic Engineering (TE) link information used for constraint-based routing.

Timing related capabilities, such as the capability for a router to perform time-stamping, and OC, TC or BC processing, need to be advertised in order for them to be taken into account during path computation. A management system or SDN controller cognizant of timing related capabilities, can prefer or even require a Timing LSP to traverse links or nodes or intefaces with the required capabilities. The optimal path will optimize the performance of the slave clock.

Extensions are required to OSPF and IS-IS in order to advertise timing related capabilities of a link. Such extensions are outside the scope of this document; however such extensions SHOULD be able to signal the following information per Router Link:

- Capable of processing PTP, NTP or other timing flows
- Capable of performing TC operation
- Capable of performing BC operation

## A.2. Signaling Extensions for Creating Timing LSPs

RSVP-TE signaling MAY be used to set up Timing LSPs. Extensions are required to RSVP-TE for this purpose. Such extensions are outside the scope of this document; however, the following information MAY be included in such extensions:

- Offset from Bottom of Stack (BoS) to the start of the Time-stamp field
- Number of VLANs in case of PW encapsulation

- Time-stamp field Type
  - Correction Field, time-stamp
- Time-stamp Field format
  - 64-bit PTPv1, 80-bit PTPv2, 32-bit NTP, 64-bit NTP, 128-bit NTP, etc.

Note that when the above optional information is signaled with RSVP-TE for a Timing LSP, all the timing packets carried in that LSP must have the same signaled characteristics. For example if time-stamp format is signaled as 64-bit PTPv1, then all timing packets must use 64-bit PTPv1 time-stamp.

## Authors' Addresses

Shahram Davari
Broadcom Corp.
San Jose, CA 95134
USA

Email: davari@broadcom.com

Amit Oren
Broadcom Corp.
San Jose, CA 95134
USA

Email: amito@broadcom.com

Manav Bhatia
Alcatel-Lucent
Bangalore
India

Email: manav.bhatia@alcatel-lucent.com

Peter Roberts
Alcatel-Lucent
Kanata
Canada

Email: peter.roberts@alcatel-lucent.com

Laurent Montini
Cisco Systems
San Jose CA
USA

Email: lmontini@cisco.com

Network Working Group
Internet Draft
Intended status: Experimental
Expires: April 2017

A. Shpiner
Mellanox
R. Tse
PMC-Sierra
C. Schelp
Oracle
T. Mizrahi
Marvell

October 27, 2016

# Multi-Path Time Synchronization
# draft-ietf-tictoc-multi-path-synchronization-07.txt

## Abstract

Clock synchronization protocols are very widely used in IP-based networks. The Network Time Protocol (NTP) has been commonly deployed for many years, and the last few years have seen an increasingly rapid deployment of the Precision Time Protocol (PTP). As time-sensitive applications evolve, clock accuracy requirements are becoming increasingly stringent, requiring the time synchronization protocols to provide high accuracy. This memo describes a multi-path approach to PTP and NTP over IP networks, allowing the protocols to run concurrently over multiple communication paths between the master and slave clocks, without modifying these protocols. The multi-path approach can significantly contribute to clock accuracy, security and fault tolerance. The multi-path approach that is presented in this document enables backward compatibility with nodes that do not support the multi-path functionality.

## 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/1id-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 April 27, 2017.



Table of Contents

## 1. Introduction

The two most common time synchronization protocols in IP networks are the Network Time Protocol [NTP], and the Precision Time Protocol (PTP), defined in the IEEE 1588 standard [IEEE1588].
The accuracy of the time synchronization protocols directly depends on the stability and the symmetry of propagation delays on both directions between the master and slave clocks. Depending on the nature of the underlying network, time synchronization protocol packets can be subject to variable network latency or path asymmetry (e.g. [ASSYMETRY], [ASSYMETRY2]). As time sensitive applications evolve, accuracy requirements are becoming increasingly stringent.

Using a single network path in a clock synchronization protocol closely ties the slave clock accuracy to the behavior of the specific path, which may suffer from temporal congestion, faults or malicious attacks. Relying on multiple clock servers as in NTP solves these problems, but requires active maintenance of multiple accurate sources in the network, which is not always possible. The usage of Transparent Clocks (TC) in PTP solves the congestion problem by eliminating the queueing time from the delay calculations, but does not address security or fault-tolerance aspects.

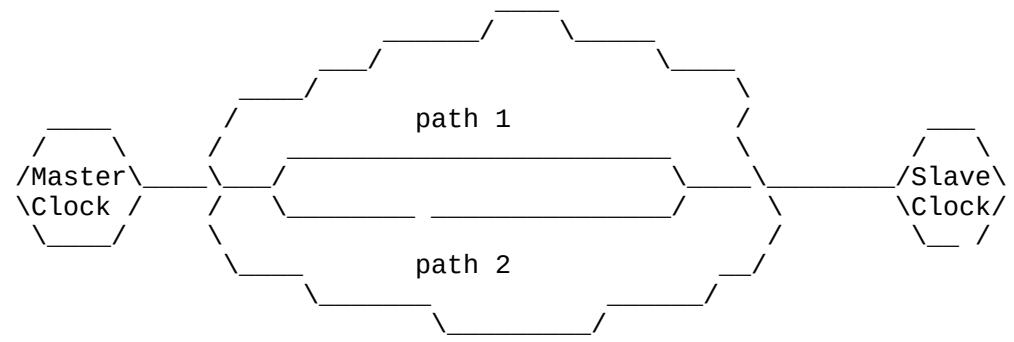



Figure 1 Multi-Path Connection

Since master and slave clocks are often connected through more than one path in the network, as shown in Figure 1, [SLAVEDIV] suggested that a time synchronization protocol can be run over multiple paths, providing several advantages. First, it can significantly increase the clock accuracy as shown in [SLAVEDIV]. Second, this approach provides additional security, allowing to mitigate man-in-the-middle attacks against the time synchronization protocol [DELAY-ATT]. Third, using multiple paths concurrently provides an inherent failure protection mechanism.

This document introduces Multi-Path PTP (MPPTP) and Multi-Path NTP (MPNTP). The functionality of the multi-path approach is defined at the network layer and does not require any changes in the PTP or in the NTP protocols.

MPPTP and MPNTP are defined over IP networks. As IP networks typically combine ECMP routing, this property is leveraged for the multiple paths used in MPPTP and MPNTP. The key property of the multi-path approach is that clocks in the network can use more than one IP address. Each {master IP, slave IP} address pair defines a path. Depending on the network topology and configuration, the IP combination pairs can form multiple diverse paths used by the multi-path synchronization protocols. It has been shown [MULTI] that using multiple IP addresses over the wide Internet indeed allows two end-points to attain multiple diverse paths.

This document introduces two variants of the multi-path approach; a variant that requires both master and slave nodes to support the multi-path functionality, referred to as the dual-ended variant, and a backward compatible variant that allows a multi-path clock to connect to a conventional single-path clock, referred to as the single-ended variant.

## 2. Conventions Used in this Document

### 2.1. Abbreviations

BMC Best Master Clock [IEEE1588]

ECMP Equal Cost Multiple Path

LAN Local Area Network

MPNTP Multi-Path Network Time Protocol

MPPTP Multi-Path Precision Time Protocol

NTP Network Time Protocol [NTP]

PTP Precision Time Protocol [IEEE1588]

## 2.2. Terminology

In the NTP terminology, a time synchronization protocol is run between a client and a server, while PTP uses the terms master and slave. Throughout this document, the sections that refer to both PTP and NTP generically use the terms master and slave.

# 3. Multiple Paths in IP Networks

## 3.1. Load Balancing

Traffic sent across IP networks is often load balanced across multiple paths. The load balancing decisions are typically based on packet header fields: source and destination addresses, Layer 4 ports, the Flow Label field in IPv6, etc.
Three common load balancing criteria are per-destination, per-flow and per-packet. The per-destination load balancers take a load balancing decision based on the destination IP address. Per-flow load balancers use various fields in the packet header, e.g., IP addresses and Layer 4 ports, for the load balancing decision. Per-packet load balancers use flow-blind techniques such as round-robin without basing the choice on the packet content.

## 3.2. Using Multiple Paths Concurrently

To utilize the diverse paths that traverse per-destination load-balancers or per-flow load-balancers, the packet transmitter can vary the IP addresses in the packet header. The analysis in [PARIS2] shows that a significant majority of the flows on the internet traverse per-destination or per-flow load-balancing. It presents statistics that 72% of the flows traverse per-destination load balancing and 39% of the flows traverse per-flow load-balancing, while only a negligible part of the flows traverse per-packet load balancing. These statistics show that the vast majority of the traffic on the internet is load balanced based on packet header fields.

The approaches in this draft are based on varying the source and destination IP addresses in the packet header. Possible extensions have been considered that also vary the UDP ports. However some of the existing implementations of PTP and NTP use fixed UDP port values in both the source and destination UDP port fields, and thus do not allow this approach.

### 3.3. Two-Way Paths

A key property of IP networks is that packets forwarded from A to B do not necessarily traverse the same path as packets from B to A. Thus, we define a two-way path for a master-slave connection as a pair of one-way paths: the first from master to slave and the second from slave to master.

If possible, a traffic engineering approach can be used to verify that time synchronization traffic is always forwarded through bidirectional two-way paths, i.e., that each two-way path uses the same route on the forward and reverse directions, thus allowing propagation time symmetry. However, in the general case two-way paths do not necessarily use the same path for the forward and reverse directions.

## 4. Solution Overview

The multi-path time synchronization protocols we present are comprised of two building blocks; one is the path configuration and identification, and the other is the algorithm used by the slave to combine the information received from the various paths.

### 4.1. Path Configuration and Identification

The master and slave clocks must be able to determine the path of transmitted protocol packets, and to identify the path of incoming protocol packets. A path is determined by a {master IP, slave IP} address pair. The synchronization protocol message exchange is run independently through each path.

Each IP address pair defines a two-way path, and thus allows the clocks to bind a transmitted packet to a specific path, or to identify the path of an incoming packet.

If possible, the routing tables across the network should be configured with multiple traffic engineered paths between the pair of clocks. By carefully configuring the routers in such networks it is possible to create diverse paths for each of the IP address pairs between two clocks in the network. However, in public and provider networks the load balancing behavior is hidden from the end users. In this case the actual number of paths may be less than the number of IP address pairs, since some of the address pairs may share common paths.

### 4.2. Combining

Various methods can be used for combining the time information received from the different paths. The output of the combining algorithm is the accurate time offset. Combining methods are further discussed in Section 6.

## 5. Multi-Path Time Synchronization over IP Networks

### 5.1. Overview

This section presents two variants of MPPTP and MPNTP; single-ended multi-path time synchronization and dual-ended multi-path time synchronization. In the first variant, the multi-path approach is only implemented by the slave and the master is not aware of its usage. In the second variant, all clocks use multiple paths.

The dual-ended variant provides higher path diversity by using multiple IP addresses at both ends, the master and slave, while the single-ended variant only uses multiple addresses at the slave. Consequently, the single-ended approach is can interoperate with existing implementations, which do not use multiple paths. The dual-ended and single-ended approaches can co-exist in the same network; each slave selects the connection(s) it wants to make with the available masters. A dual-ended slave could switch to single-ended mode if it does not see any dual-ended masters available. A single-ended slave could connect to a single IP address of a dual-ended master.

Multi-path time synchronization, in both variants, requires clocks to use multiple IP addresses. Using multiple IP addresses introduces a tradeoff. A large number of IP addresses allows a large number of diverse paths, providing the advantages of slave diversity discussed in Section 1. On the other hand, a large number of IP addresses is more costly, requires the network topology to be more redundant, and exacts extra management overhead.

If possible, the set of IP addresses for each clock should be chosen in a way that enables the establishment of paths that are the most different. If the load balancing rules in the network are known, it is possible to choose the IP addresses in a way that enforces path diversity. However, even if the load balancing scheme is not known, a careful choice of the IP addresses can increase the probability of path diversity. For example, choosing multiple addresses with different prefixes is likely to produce higher path diversity, as BGP routers are more likely to route these different prefixes through different routes.

The use of Network Address Translation (NAT) may significantly reduce the effectiveness of multi-path synchronization in some cases. For example, if a master uses multiple IP addresses that are translated to a single IP address, the path diversity can be dramatically reduced compared to a network that does not use NAT. Thus, path discovery should be used to identify the possible paths between the master and slave. Path discovery is further discussed in Section 5.4.

The concept of using multiple IP addresses or multiple interfaces is a well-established concept that is being used today by various applications and protocols, e.g., [MPTCP]. Using multiple interfaces introduces some challenges and issues, which were presented and discussed in [MIF].

The descriptions in this section refer to the end-to-end scheme of PTP, but are similarly applicable to the peer-to-peer scheme. MPNTP, as described in this document, refers to the NTP client-server mode, although the concepts described here can be extended to include the symmetric variant as well.

Multi-path synchronization by nature requires protocol messages to be sent as unicast. Specifically in PTP, the following messages must be sent as unicast in MPPTP: Sync, Delay_Req, Delay_Resp, PDelay_Req, PDelay_Resp, Follow_Up, and PDelay_Resp_Follow_Up. Note that [IEEE1588] allows these messages to be sent either as multicast or as unicast.

## 5.2. Single-Ended Multi-Path Synchronization

In the single-ended approach, only the slave is aware of the fact that multiple paths are used, while the master is agnostic to the usage of multiple paths. This approach allows a hybrid network, where some of the clocks are multi-path clocks, and others are conventional one-path clocks. A single-ended multi-path clock presents itself to the network as N independent clocks, using N IP addresses, as well as N clock identity values (in PTP). Thus, the usage of multiple slave identities by a slave clock is transparent from the master’s point of view, such that it treats each of the identities as a separate slave clock.

### 5.2.1. Single-Ended MPPTP Synchronization Message Exchange

The single-ended MPPTP message exchange procedure is as follows.

- Each single-ended MPPTP clock has a fixed set of N IP addresses and N corresponding clockIdentities. Each clock arbitrarily defines one of its IP addresses and clockIdentity values as the clock primary identity.

- A single-ended MPPTP port sends Announce messages only from its primary identity, according to the BMC algorithm.

- The BMC algorithm at each clock determines the master, based on the received Announce messages.

- A single-ended MPPTP port that is in the ’slave’ state uses unicast negotiation to request the master to transmit unicast messages to each of the N slave clock identities. The slave port periodically sends N Signaling messages to the master, using each of its N identities. The Signaling message includes the REQUEST_UNICAST_TRANSMISSION_TLV.

- The master periodically sends unicast Sync messages from its primary identity, identified by the sourcePortIdentity and IP address, to each of the slave identities.

- The slave, upon receiving a Sync message, identifies its path according to the destination IP address. The slave sends a Delay_Req unicast message to the primary identity of the master. The Delay_Req is sent using the slave identity corresponding to the path the Sync was received through. Note that the rate of Delay_Req messages may be lower than the Sync message rate, and thus a Sync message is not necessarily followed by a Delay_Req.

- The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the IP address and sourcePortIdentity from the Delay_Req message.

- Upon receiving the Delay_Resp message, the slave identifies the path using the destination IP address and the requestingPortIdentity. The slave can then compute the corresponding path delay and the offset from the master.

- The slave combines the information from all negotiated paths.

#### 5.2.2. Single-Ended MPNTP Synchronization Message Exchange

The single-ended MPNTP message exchange procedure is as follows.

- o A single-ended MPNTP client has N separate identities, i.e., N IP addresses. The assumption is that the server information, including its IP address is known to the NTP clients. This is a fair assumption, as typically the address(es) of the NTP server(s) are provided to the NTP client by configuration.

- o A single-ended MPNTP client initiates the NTP protocol with an NTP server N times, using each of its N identities.

- o The NTP protocol is maintained between the server and each of the N client identities.

- o The client sends NTP messages to the master using each of its N identities.

- o The server responds to the client’s NTP messages using the IP address from the received NTP packet.

- o The client, upon receiving an NTP packet, uses the IP destination address to identify the path it came through, and uses the time information accordingly.

- o The client combines the information from all paths.

### 5.3. Dual-Ended Multi-Path Synchronization

In dual-ended multi-path synchronization each clock has N IP addresses. Time synchronization messages are exchanged between some of the combinations of {master IP, slave IP} addresses, allowing multiple paths between the master and slave. Note that the actual number of paths between the master and slave may be less than the number of chosen {master, slave} IP address pairs.

Once the multiple two-way connections are established, a separate synchronization protocol exchange instance is run through each of them.

#### 5.3.1. Dual-Ended MPPTP Synchronization Message Exchange

The dual-ended MPPTP message exchange procedure is as follows.

- o Every clock has N IP addresses, but uses a single clockIdentity.

- o The BMC algorithm at each clock determines the master. The master is identified by its clockIdentity, allowing other clocks to know the multiple IP addresses it uses.

- When a clock sends an Announce message, it sends it from each of its IP addresses with its clockIdentity.

- A dual-ended MPPTP port that is in the ’slave’ state uses unicast negotiation to request the master to transmit unicast messages to some or all of its N_s IP addresses. This negotiation is done individually between a slave IP address and the corresponding master IP address that the slave desires a connection with. The slave port periodically sends Signaling messages to the master, using some or all of its N_s IP addresses as source, to the corresponding master’s N_m IP addresses. The Signaling message includes the REQUEST_UNICAST_TRANSMISSION_TLV.

- The master periodically sends unicast Sync messages from each of its IP addresses to the corresponding slave IP addresses for which a unicast connection was negotiated.

- The slave, upon receiving a Sync message, identifies its path according to the {source, destination} IP addresses. The slave sends a Delay_Req unicast message, swapping the source and destination IP addresses from the Sync message. Note that the rate of Delay_Req messages may be lower than the Sync message rate, and thus a Sync message is not necessarily followed by a Delay_Req.

- The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the sourcePortIdentity from the Delay_Req message, and swapping the IP addresses from the Delay_Req.

- Upon receiving the Delay_Resp message, the slave identifies the path using the {source, destination} IP address pair. The slave can then compute the corresponding path delay and the offset from the master.

- The slave combines the information from all negotiated paths.

#### 5.3.2. Dual-Ended MPNTP Synchronization Message Exchange

The MPNTP message exchange procedure is as follows.

- Each NTP clock has a set of N IP addresses. The assumption is that the server information, including its multiple IP addresses is known to the NTP clients.

- The MPNTP client chooses N_svr of the N server IP addresses and N_c of the N client IP addresses and initiates the N_svr*N_c instances of the protocol, one for each {server IP, client IP} pair, allowing the client to combine the information from the N_s*N_c paths.
  (N_svr and N_c indicate the number of IP addresses of the server and client, respectively, which a client chooses to connect with)

- The client sends NTP messages to the master using each of the source-destination address combinations.

- The server responds to the client's NTP messages using the IP address combination from the received NTP packet.

- Using the {source, destination} IP address pair in the received packets, the client identifies the path, and performs its computations for each of the paths accordingly.

- The client combines the information from all paths.

### 5.4. Using Traceroute for Path Discovery

The approach described thus far uses multiple IP addresses in a single clock to create multiple paths. However, although each two-way path is defined by a different {master, slave} address pair, some of the IP address pairs may traverse exactly the same network path, making them redundant.

Traceroute-based path discovery can be used for filtering only the IP addresses that obtain diverse paths. 'Paris Traceroute' [PARIS] and 'TraceFlow' [TRACEFLOW] are examples of tools that discover the paths between two points in the network. It should be noted that this filtering approach is effective only if the Traceroute implementation uses the same IP addresses and UDP ports as the synchronization protocol packets. Since some Traceroute implementations vary the UDP ports, they may not be effective in identifying and filtering redundant paths in synchronization protocols.

The Traceroute-based filtering can be implemented by both master and slave nodes, or it can be restricted to run only on slave nodes to reduce the overhead on the master. For networks that guarantee that the path of the timing packets in the forward and reverse direction are the same, path discovery should only be performed at the slave.

Since network routes change over time, path discovery and redundant path filtering should be performed periodically. Two {master,slave} pairs that produce two diverse paths may be rerouted to use the same

paths. Thus, the set of addresses that are used by each clock should be reassessed regularly.

### 5.5. Using Unicast Discovery for MPPTP

As presented above, MPPTP uses Announce messages and the BMC algorithm to discover the master. The unicast discovery option of PTP can be used as an alternative.

When using unicast discovery the MPPTP slave ports maintain a list of the IP addresses of the master. The slave port uses unicast negotiation to request unicast service from the master, as follows:

- In single-ended MPPTP, the slave uses unicast negotiation from each of its identities to the master's (only) identity.
- In dual-ended MPPTP, the slave uses unicast negotiation from its IP addresses, each to a corresponding master IP address to request unicast synchronization messages.

Afterwards, the message exchange continues as described in sections 5.2.1. and 5.3.1.

The unicast discovery option can be used in networks that do not support multicast or in networks in which the master clocks are known in advance. In particular, unicast discovery avoids multicasting Announce messages.

## 6. Combining Algorithm

Previous sections discussed the methods of creating the multiple paths and obtaining the time information required by the slave algorithm. Once the time information is received through each of the paths, the slave should use a combining algorithm, which consolidates the information from the different paths into a single clock. Various methods have been suggested for combining information from different paths or from different clocks, e.g., [NTP], [SLAVEDIV], [HIGH-AVAI], [KALMAN]. The choice of the combining algorithm is local to the slave, and does not affect interoperability. Hence, this document does not define a specific method to be used by the slave. The combining algorithm should be chosen carefully based on the system properties, as different combining algorithms provide different advantages. For example, some combining algorithms (e.g., [NTP], [DELAY-ATT]) are intended to be robust in the face of security attacks, while other combining algorithms (e.g., [KALMAN]) are more resilient to random delay variation.

## 7. Security Considerations

The security aspects of time synchronization protocols are discussed in detail in [RFC7384]. The methods described in this document propose to run a time synchronization protocol through redundant paths, and thus allow to detect and mitigate man-in-the-middle attacks, as described in [DELAY-ATT]. Specifically, multi-path synchronization can mitigate the following threats (as per [RFC7384]):

- Packet manipulation (Section 3.2.1 of [RFC7384]).
- Packet Interception and Removal (Section 3.2.5 of [RFC7384]).
- Packet delay manipulation (Section 3.2.6 of [RFC7384]).

It should be noted that when using multiple paths, these paths may partially overlap, and thus an attack that takes place in a common segment of these paths is not mitigated by the redundancy. Moreover, an on-path attacker may in some cases have access to more than one router, or may be able to migrate from one router to another. Therefore, when using multiple paths it is important for the paths to be as diverse and as independent as possible, making the redundancy scheme more tolerant to on-path attacks.

It should be noted that the multi-path approach requires the master (or NTP server) to dedicate more resources to each slave (client) than the conventional single-path approach. Hence, well-known Distributed Denial-of-Service (DDoS) attacks may porentially be amplified when the multi-path approach is enabled.

## 8. Scope of the Experiment

This memo is published as an experimental RFC. The purpose of the experimental period is to allow the community to analyze and to verify the methods defined in this document. An experimental evaluation of some of these methods has been published in [MULTI]. It is expected that the experimental period will allow the methods to be further investigated and verified by the community. The duration of the experiment is expected to be no less than two years from the publication of this document.

## 9. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

## 10. Acknowledgments

The authors gratefully acknowledge the useful comments provided by Peter Meyer, Doug Arnold, Joe Ableу, Zhen Cao, Watson Ladd, and Mirja Kuehlewind, as well as other comments received from the TICTOC working group participants.

This document was prepared using 2-Word-v2.0.template.dot.

## 11. References

### 11.1. Normative References

[IEEE1588] IEEE Instrumentation and Measurement Society, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588, 2008.

[NTP] D. Mills, J. Martin, J. Burbank, W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", IETF, RFC 5905, 2010.

### 11.2. Informative References

[ASSYMETRY] Yihua He and Michalis Faloutsos and Srikanth Krishnamurthy and Bradley Huffaker, "On routing asymmetry in the internet", IEEE Globecom, 2005.

[ASSYMETRY2] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao, "A measurement study of internet delay asymmetry", PAM’08, 2008.

[DELAY-ATT] T. Mizrahi, "A Game Theoretic Analysis of Delay Attacks against Time Synchronization Protocols", ISPCS, 2012.

[HIGH-AVAI] P. Ferrari, A. Flammini, S. Rinaldi, G. Prytz, "High availability IEEE 1588 nodes over IEEE 802.1 aq Shortest Path Bridging networks" ISPCS, 2013.

[KALMAN] G. Giorgi, C. Narduzzi, "Kalman filtering for multi-path network synchronization" ISPCS, 2014.

[MIF] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, DOI 10.17487/RFC6418, November 2011, <http://www.rfc-editor.org/info/rfc6418>.

[MPTCP] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[MULTI] A. Shpiner, Y. Revah, T. Mizrahi, "Multi-path Time Protocols" ISPCS, 2013.

[PARIS] Brice Augustin, Timur Friedman and Renata Teixeira, "Measuring Load-balanced Paths in the Internet", IMC, 2007.

[PARIS2] B. Augustin, T. Friedman, and R. Teixeira, "Measuring Multipath Routing in the Internet", IEEE/ACM Transactions on Networking, 19(3), p. 830 - 840, June 2011.

[RFC7384] T. Mizrahi, "Security Requirements of Time Protocols in Packet Switched Networks", IETF, RFC 7384, 2014.

[SLAVEDIV] T. Mizrahi, "Slave Diversity: Using Multiple Paths to Improve the Accuracy of Clock Synchronization Protocols", ISPCS, 2012.

[TRACEFLOW] J. Narasimhan, B. V. Venkataswami, R. Groves and P. Hoose, "Traceflow", IETF, draft-janapath-intarea-traceflow, work in progress, 2012.

Authors' Addresses

Alex Shpiner
Mellanox Technologies, Ltd.
Hakidma 26
Ofer Industrial Park
Yokneam, 2069200, Israel

Email: alexshp@mellanox.com

Richard Tse
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: Richard.Tse@pmcs.com

Craig Schelp
Oracle

Email: craig.schelp@gmail.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692, Israel

Email: talmi@marvell.com

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

D. Arnold
Meinberg-USA
H. Gerstung
Meinberg

December 11, 2020

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

Abstract

This document describes a profile for the use of the Precision Time Protocol in an IPV4 or IPv6 Enterprise information system environment. The profile uses the End to End Delay Measurement Mechanism, allows both multicast and unicast Delay Request and Delay Response Messages.

Status of This Memo

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

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

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

This Internet-Draft will expire on June 14, 2021.

Table of Contents



## 1. Introduction

The Precision Time Protocol ("PTP"), standardized in IEEE 1588, has been designed in its first version (IEEE 1588-2002) with the goal to minimize configuration on the participating nodes. Network communication was based solely on multicast messages, which unlike NTP did not require that a receiving node ("slave clock") in IEEE 1588-2008 [IEEE1588] needs to know the identity of the time sources in the network (the Master Clocks).

The "Best Master Clock Algorithm" (IEEE 1588-2008 [IEEE1588] Subclause 9.3), a mechanism that all participating PTP nodes must follow, set up strict rules for all members of a PTP domain to determine which node shall be the active sending time source (Master Clock). Although the multicast communication model has advantages in smaller networks, it complicated the application of PTP in larger networks, for example in environments like IP based telecommunication networks or financial data centers. It is considered inefficient that, even if the content of a message applies only to one receiver, it is forwarded by the underlying network (IP) to all nodes,

requiring them to spend network bandwidth and other resources, such as CPU cycles, to drop the message.

The second revision of the standard (IEEE 1588-2008) is the current version (also known as PTPv2) and introduced the possibility to use unicast communication between the PTP nodes in order to overcome the limitation of using multicast messages for the bi-directional information exchange between PTP nodes. The unicast approach avoided that, in PTP domains with a lot of nodes, devices had to throw away more than 99% of the received multicast messages because they carried information for some other node. PTPv2 also introduced PTP profiles (IEEE 1588-2008 [IEEE1588] subclause 19.3). This construct allows organizations to specify selections of attribute values and optional features, simplifying the configuration of PTP nodes for a specific application. Instead of having to go through all possible parameters and configuration options and individually set them up, selecting a profile on a PTP node will set all the parameters that are specified in the profile to a defined value. If a PTP profile definition allows multiple values for a parameter, selection of the profile will set the profile-specific default value for this parameter. Parameters not allowing multiple values are set to the value defined in the PTP profile. Many PTP features and functions are optional, and a profile should also define which optional features of PTP are required, permitted, or prohibited. It is possible to extend the PTP standard with a PTP profile by using the TLV mechanism of PTP (see IEEE 1588-2008 [IEEE1588] subclause 13.4), defining an optional Best Master Clock Algorithm and a few other ways. PTP has its own management protocol (defined in IEEE 1588-2008 [IEEE1588] subclause 15.2) but allows a PTP profile specify an alternative management mechanism, for example SNMP.

## 2. Requirements Language

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

## 3. Technical Terms

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

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

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

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

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

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

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

- Domain: Every PTP message contains a domain number. Domains are treated as separate PTP systems in the network. Clocks, however, can combine the timing information derived from multiple domains.

- End to End Delay Measurement Mechanism: A network delay measurement mechanism in PTP facilitated by an exchange of messages between a Master Clock and Slave Clock.

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

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

- Master Clock: a clock with at least one port in the master state.

- NTP: Network Time Protocol, defined by RFC 5905, see RFC 5905 [RFC5905]

- Ordinary Clock: A clock that has a single Precision Time Protocol (PTP) port in a domain and maintains the timescale used in the domain. It may serve as a Master Clock, or be a slave clock.

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

- Preferred Master: A device intended to act primarily as the Grandmaster of a PTP system, or as a back up to a Grandmaster.

- PTP: The Precision Time Protocol, the timing and synchronization protocol defined by IEEE 1588.

- PTP port: An interface of a PTP clock with the network. Note that there may be multiple PTP ports running on one physical interface, for example, a unicast slave which talks to several Grandmaster clocks in parallel.

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

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

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

- Slave Only Clock: An Ordinary Clock which cannot become a Master Clock.

- TLV: Type Length Value, a mechanism for extending messages in networked communications.

- Transparent Clock. A device that measures the time taken for a PTP event message to transit the device and then updates the message with a correction for this transit time.

- Unicast Discovery: A mechanism for PTP slaves to establish a unicast communication with PTP masters using a configures table of master IP addresses and Unicast Message Negotiation.

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

## 4. Problem Statement

This document describes a version of PTP intended to work in large enterprise networks. Such networks are deployed, for example, in financial corporations. It is becoming increasingly common in such networks to perform distributed time tagged measurements, such as one-way packet latencies and cumulative delays on software systems spread across multiple computers. Furthermore, there is often a desire to check the age of information time tagged by a different

machine. To perform these measurements, it is necessary to deliver a common precise time to multiple devices on a network. Accuracy currently required in the Financial Industry range from 100 microseconds to 100 nanoseconds to the Grandmaster. This profile does not specify timing performance requirements, but such requirements explain why the needs cannot always be met by NTP, as commonly implemented. Such accuracy cannot usually be achieved with a traditional time transfer such as NTP, without adding non-standard customizations such as hardware time stamping, and on path support. These features are currently part of PTP, or are allowed by it. Because PTP has a complex range of features and options it is necessary to create a profile for enterprise networks to achieve interoperability between equipment manufactured by different vendors.

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

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

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

## 5. Network Technology

This PTP profile SHALL operate only in networks characterized by UDP RFC 768 [RFC0768] over either IPv4 RFC 791 [RFC0791] or IPv6 RFC 8200 [RFC8200], as described by Annexes D and E in IEEE 1588 [IEEE1588] respectively. If a network contains both IPv4 and IPv6, then they SHALL be treated as separate communication paths. Clocks which communicate using IPv4 can interact with clocks using IPv6 if there is an intermediary device which simultaneously communicates with both IP versions. A Boundary Clock might perform this function, for example. A PTP domain SHALL use either IPv4 or IPv6 over a communication path, but not both. The PTP system MAY include switches and routers. These devices MAY be Transparent Clocks, boundary Clocks, or neither, in any combination. PTP Clocks MAY be Preferred Masters, Ordinary Clocks, or Boundary Clocks. The Ordinary Clocks may be Slave Only Clocks, or be master capable.

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

since Transparent Clocks are required to change the source address of any packet which they alter. In IPv4 networks some clocks might be hidden behind a NAT, which hides their IP addresses from the rest of the network. Note also that the use of NATs may place limitations on the topology of PTP networks, depending on the port forwarding scheme employed. Details of implementing PTP with NATs are out of scope of this document.

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

## 6. Time Transfer and Delay Measurement

Master Clocks, Transparent Clocks and Boundary Clocks MAY be either one-step clocks or two-step clocks. Slave clocks MUST support both behaviors. The End to End Delay Measurement Method MUST be used.

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

Sync messages MUST be sent as PTP event multicast messages (UDP port 319) to the PTP primary IP address. Two step clocks SHALL send Follow-up messages as PTP general messages (UDP port 320). Announce messages MUST be sent as multicast messages (UDP port 320) to the PTP primary address. The PTP primary IP address is 224.0.1.129 for IPv4 and FF0X:0:0:0:0:0:0:181 for Ipv6, where X can be a value between 0x0 and 0xF, see IEEE 1588 [IEEE1588] Annex E, Section E.3.

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

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

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

## 7. Default Message Rates

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

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

## 8. Requirements for Master Clocks

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

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

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

## 9. Requirements for Slave Clocks

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

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

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

## 10. Requirements for Transparent Clocks

Transparent Clocks SHALL NOT change the transmission mode of an Enterprise Profile PTP message. For example, a Transparent Clock SHALL NOT change a unicast message to a multicast message. Transparent Clocks SHOULD support multiple domains. Transparent Clocks which syntonize to the master clock will need to maintain separate clock rate offsets for each of the supported domains.

## 11. Requirements for Boundary Clocks

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

## 12. Management and Signaling Messages

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

## 13. Forbidden PTP Options

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

## 14. Interoperation with IEEE 1588 Default Profile

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

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

## 15. Profile Identification

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

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

This profile was specified by the IETF

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

## 16. Acknowledgements

The authors would like to thank members of IETF for reviewing and providing feedback on this draft.

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

## 17. IANA Considerations

There are no IANA requirements in this specification.

## 18. Security Considerations

Protocols used to transfer time, such as PTP and NTP can be important to security mechanisms which use time windows for keys and authorization. Passing time through the networks poses a security risk since time can potentially be manipulated. The use of multiple simultaneous masters, using multiple PTP domains can mitigate problems from rogue masters and man-in-the-middle attacks. See sections 9 and 10. Additional security mechanisms are outside the scope of this document.

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

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

## 19. References

### 19.1. Normative References

[IEEE1588] Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2008, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems."", 7 2008, <https://www.ieee.org>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

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

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

## 19.2. Informative References

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

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <https://www.rfc-editor.org/info/rfc7384>.

## Authors' Addresses

Doug Arnold
Meinberg-USA
3 Concord Rd
Shrewsbury, Massachusetts 01545
USA

Email: doug.arnold@meinberg-usa.com

Heiko Gerstung
Meinberg
Lange Wand 9
Bad Pyrmont 31812
Germany

Email: heiko.gerstung@meinberg.de

TICTOC Working Group
INTERNET DRAFT
Intended status: Standards Track

Vinay Shankarkumar
Laurent Montini
Cisco Systems

Tim Frost
Calnex Solutions Ltd.

Greg Dowd
Microsemi

Expires: September 17, 2017

March 17, 2017

# Precision Time Protocol Version 2 (PTPv2) Management Information Base
# draft-ietf-tictoc-ptp-mib-12.txt

## Status of this Memo



## Copyright Notice

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines objects for managing networks using Precision Time Protocol (PTP), specified in IEEE Std. 1588(TM)-2008.

This memo specifies a MIB module in a manner that is both compliant to the SMIv2, and semantically identical to the peer SMIv1 definitions.

Table of Contents



## 1. Introduction

This memo defines a portion of the Management Information Base (MIB) module for use with network management protocols in the Internet Community. In particular, it describes managed objects used for managing PTP devices including the ordinary clock, transparent clock, boundary clocks.

This MIB module is restricted to reading standard PTP data elements, as described in [IEEE 1588-2008]. This enables it to monitor the operation of PTP clocks within the network. It is envisioned this MIB module will complement other managed objects to be defined that will provide more detailed information on the performance of PTP

clocks supporting the Telecom Profile defined in [G.8265.1], and any future profiles that may be defined. Those objects are considered out of scope for the current draft.

Similarly, this MIB module is read-only and not intended to provide the ability to configure PTP clocks. Since PTP clocks are often embedded in other network elements such as routers, switches and gateways, this ability is generally provided via the configuration interface for the network element.

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].

### 1.1. Relationship to other Profiles and MIBs

This MIB module is intended to be used with the default PTP profile described in [IEEE 1588-2008] when running over the IP network layer. As stated above, it is envisioned this MIB module will complement other managed objects to be defined to monitor and measure the performance of PTP clocks supporting specific PTP profiles, e.g. the Telecom Profile defined in [G.8265.1].

Some other PTP profiles have their own MIB modules defined as part of the profile, and this MIB module is not intended to replace those MIB modules.

### 1.2. Change Log

This section tracks changes made to the revisions of the Internet Drafts of this document. It will be *deleted* when the document is published as an RFC.

draft-vinay-tictoc-ptp-mib

-00 Mar 11 Initial version; showed structure of MIB

draft-ietf-tictoc-ptp-mib

-00 Jul 11 First full, syntactically correct and compileable MIB

-01 Jan 12 Revised following comments from Bert Wijnen:
- revised introduction to clarify the scope, and the relationship to other MIBs and profiles
- changed name to "ptpbase"
- corrected some data types
- corrected references and typos

-02 Jul 12 Revised following comment at IETF83:

- changed "ptpbaseClockPortRunningIPversion" to the more generic "ptpbaseClockPortRunningTransport", covering all transport types defined in [IEEE 1588-2008] (i.e. IPv4, IPv6, Ethernet, DeviceNet and ControlNet).
- changed addresses associated with transports from "InetAddress" (for the IP transport) to a string, to allow for the different transport types.

-03 Jul 12 Minor changes following comments from Andy Bierman:
- corrected some compilation errors
- moved OBJECT-GROUP and MODULE-COMPLIANCE macros to the end

-04 Jan 13 Changes:
- Use of 'AutonomousType' import
- Display hint being specified for ClockIdentity, ClockInterval, ClockPortTransportTypeAddress Textual Conventions
- Removal of the Textual convention ClockPortTransportType, replaced with the wellKnownTransportTypes
- Modified ptpbaseClockPortCurrentPeerAddressType, ptpbaseClockPortRunningTransport, ptpbaseClockPortAssociateAddressType, to use AutonomousType.
- various textual changes to descriptive text in response to comments

-05 Feb 13 Several changes in response to comments from Alun Luchuk and Kevin Gross:
- Modified the use of wellKnownTransportTypes and wellKnownEncapsulationTypes
- changed ptpbaseClockPortSyncOneStep to ptpbaseClockPortSyncTwoStep to match [IEEE 1588-2008] semantics
- Re-ordered textual conventions to be alphabetic
- Changed some types from Integer32 to use defined textual conventions
- various minor descriptive text changes

-06 Mar 14 Updated author information, and fixed typos

-07 Mar 15 Updated author information, and fixed typo/enum

-08 Feb 16 Updated MIB in response to Brian Haberman's comments:
- Fixed MIB date
- Fixed references to [IEEE 1588-2008]
- Changed "router" for "node"

-09 Apr 16 Updated following Dan Romascanu's MIB Doctor comments

-10 Aug 16 Update following further feedback from Dan Romascanu. Also updated security section to list out all objects with MAX-ACCESS other than non-accessible, in response to comments from Deborah Brungard and Alissa Cooper.

-11 Aug 16 Used corrected version of MIB text
- Reduced the DESCIPTION section and moved to section 3
- Added clarification that PtpClockIdentity can also be non-EUI-64 address
- Clarifications on PtpClockPortTransportTypeAddress, and mentioned counters being discontinuous
- Made PtpClockQualityClassType as enumeration
Updated overview section with a longer description.

-12 Mar 17 Replaced direct quotations of [IEEE 1588-2008] with references to avoid copyright issues.

## 2. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

o An overall architecture, described in STD62, [RFC 3411].

o Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and described in STD 16: [RFC 1155], [RFC 1212] and [RFC 1215].
The second version, called SMIv2, is described in STD 58: [RFC 2578], [RFC 2579] and [RFC 2580].

o Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15 [RFC 1157]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in [RFC 1901] and [RFC 1906]. The third version of the message protocol is called SNMPv3 and described in STD62: [RFC 3417], [RFC 3412] and [RFC 3414].

o Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15 [RFC 1157]. A second set of protocol operations and associated PDU formats is described in STD 62 [RFC 3416].

o A set of fundamental applications described in STD 62 [RFC 3413]

and the view-based access control mechanism described in STD 62 [RFC 3415].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This memo specifies a MIB module that is compliant to the SMIv2. A MIB module conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (e.g., use of Counter64). Some machine readable information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB module.

## 3. Overview

The objects defined in this MIB module are to be used when describing the Precision Time Protocol (PTP), as defined in [IEEE 1588-2008].

Section 6 of [IEEE 1588-2008] provides an overview of synchronization networks using PTP.

Terms used in this document have meanings as defined in section 3.1 of [IEEE 1588-2008].

## 4. IETF PTP MIB Definition

```
PTPBASE-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    OBJECT-IDENTITY,
    Gauge32,
    Unsigned32,
    Counter32,
    Counter64,
    mib-2,
    Integer32
        FROM SNMPv2-SMI
    OBJECT-GROUP,
    MODULE-COMPLIANCE
        FROM SNMPv2-CONF
    TEXTUAL-CONVENTION,
    TruthValue,
    DisplayString,
```

```
    AutonomousType
        FROM SNMPv2-TC
    InterfaceIndexOrZero
        FROM IF-MIB;


ptpbaseMIB MODULE-IDENTITY
    LAST-UPDATED    "201703120000Z"
    ORGANIZATION    "TICTOC Working Group"
    CONTACT-INFO
          "WG Email: tictoc@ietf.org

           Vinay Shankarkumar
           Cisco Systems,
           Email: vinays@cisco.com

           Laurent Montini,
           Cisco Systems,
           Email: lmontini@cisco.com

           Tim Frost,
           Calnex Solutions Ltd.,
           Email: tim.frost@calnexsol.com

           Greg Dowd,
           Microsemi Inc.,
           Email: greg.dowd@microsemi.com"

    DESCRIPTION
        "The MIB module for PTP version 2 (IEEE Std. 1588(TM)-2008)

        Overview of PTP version 2 (IEEE Std. 1588(TM)-2008)

        [IEEE 1588-2008] defines a protocol enabling precise
        synchronization of clocks in measurement and control systems
        implemented with packet-based networks, the Precision Time
        Protocol Version 2 (PTPv2).  This MIB module does not address
        the earlier version IEEE Std. 1588(TM)-2002 (PTPv1).  The
        protocol is applicable to network elements communicating using
        IP.  The protocol enables heterogeneous systems that include
        clocks of various inherent precision, resolution, and stability
        to synchronize to a grandmaster clock.

        The protocol supports system-wide synchronization accuracy in
        the sub-microsecond range with minimal network and local clock
        computing resources. [IEEE 1588-2008] uses UDP/IP or
        Ethernet and can be adapted to other mappings. It includes
        formal mechanisms for message extensions, higher sampling rates,
        correction for asymmetry, a clock type to reduce error
```

accumulation in large topologies, and specifications on how to incorporate the resulting additional data into the synchronization protocol. The [IEEE 1588-2008] defines conformance and management capability also.

MIB description

This MIB module supports the Precision Time Protocol version 2 (PTPv2, hereafter designated as PTP) features of network element system devices, when using the default PTP profile described in [IEEE 1588-2008] when running over the IP network layer.

It is envisioned this MIB module will complement other managed objects to be defined to monitor and measure the performance of the PTP devices and telecom clocks supporting specific PTP profiles.

Some other PTP profiles have their own MIB modules defined as part of the profile, and this MIB module is not intended to replace those MIB modules.

Technical terms used in this module are defined in [IEEE 1588-2008].

The MIB module refers to the sections of [IEEE 1588-2008].

Acronyms:

| | |
|---|---|
| ARB | Arbitrary Timescale |
| E2E | End-to-End |
| EUI | Extended Unique Identifier |
| GPS | Global Positioning System |
| IANA | Internet Assigned Numbers Authority |
| IP | Internet Protocol |
| MAC | Media Access Control according to [IEEE 802.3-2008] |
| MAC-48 | Used to identify hardware instances within 802-based networking applications. This is obsolete now. |
| NIST | National Institute of Standards and Technology |
| NTP | Network Time Protocol (see IETF [RFC 5905]) |
| OUI | Organizational Unique Identifier (allocated by the IEEE) |
| P2P | Peer-to-Peer |
| PTP | Precision Time Protocol |
| TAI | International Atomic Time |
| TC | Transparent Clock |
| UDP | User Datagram Protocol |
| UTC | Coordinated Universal Time |

References:

[IEEE 1588-2008] IEEE Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std. 1588(TM)-2008, 24 July 2008.

The below table specifies the object formats of the various textual conventions used.

| Data type mapping | Textual Convention | SYNTAX |
|---|---|---|
| 5.3.2 TimeInterval | PtpClockTimeInterval | OCTET STRING(SIZE(1..255)) |
| 5.3.3 Timestamp | PtpClockTimestamp | OCTET STRING(SIZE(6)) |
| 5.3.4 ClockIdentity | PtpClockIdentity | OCTET STRING(SIZE(8)) |
| 5.3.5 PortIdentity | PtpClockPortNumber | INTEGER(1..65535) |
| 5.3.7 ClockQuality | PtpClockQualityClassType | |

"

```
      -- revision log
      REVISION        "201703120000Z"
      DESCRIPTION     "Draft 12, for IESG approval remvoed the IEEE
standard texts."

      REVISION        "201608240000Z"
      DESCRIPTION     "Draft 11, for IESG approval after all comments,
including the correct MIB."

      REVISION        "201608220000Z"
      DESCRIPTION     "Draft 10, for IESG approval after all comments
addressed."

      REVISION        "201604200000Z"
      DESCRIPTION     "Draft 9, for IESG approval."

      REVISION        "201602220000Z"
      DESCRIPTION     "Draft 8, for IETF last call."

      ::= { mib-2 XXX } -- XXX to be assigned by IANA


-- Textual Conventions

PtpClockDomainType ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "The Domain is identified by an integer, the domainNumber, in
        the range of 0 to 255. An integer value that is used to assign
        each PTP device to a particular domain."
```

```
    REFERENCE   "Section 7.1 Domains, Table 2 of [IEEE 1588-2008]"
    SYNTAX      Unsigned32 (0..255)

PtpClockIdentity ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "255a"
    STATUS          current
    DESCRIPTION
        "The clock Identity is an 8-octet array and will be presented in
        the form of a character array.  Network byte order is assumed.

        The value of the PtpClockIdentity should be taken from the
        IEEE EUI-64 individual assigned numbers as indicated in
        Section 7.5.2.2.2 of [IEEE 1588-2008]. It can also be non-EUI-64
        address as defined in section 7.5.2.2.3 of [IEEE 1588-2008].

        The clock identifier can be constructed from existing EUI-48
        assignments and here is an abbreviated example extracted from
        section 7.5.2.2.2 [IEEE 1588-2008]."

    REFERENCE       "Section 7.5.2.2.1 of [IEEE 1588-2008]"
    SYNTAX          OCTET STRING (SIZE (8))

PtpClockInstanceType ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "The instance of the Clock of a given clock type in a given
        domain."
    SYNTAX          Unsigned32 (0..255)

PtpClockIntervalBase2 ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "The interval included in message types Announce, Sync,
        Delay_Req, and Pdelay_Req as indicated in section 7.7.2.1 of
        [IEEE 1588-2008]."

    REFERENCE   "Section 7.7.2.1 General interval specification of
                 [IEEE 1588-2008]"
    SYNTAX      Integer32 (-128..127)

PtpClockMechanismType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The clock type based on whether end-to-end or peer-to-peer
        mechanisms are used. The mechanism used to calculate the Mean
        Path Delay as indicated in Table 9 of [IEEE 1588-2008]."
```

```
    REFERENCE
        "Sections 8.2.5.4.4 portDS.delayMechanism,
        6.6.4  Measuring link propagation delay in clocks supporting
        peer-to-peer path correction,
        7.4.2 communication Path asymmetry of [IEEE 1588-2008]."
    SYNTAX      INTEGER {
                    e2e(1),
                    p2p(2),
                    disabled(254)
                    }

PtpClockPortNumber ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "An index identifying a specific Precision Time Protocol (PTP)
        port on a PTP node."

    REFERENCE
        "Sections 7.5.2.3 portNumber and 5.3.5 PortIdentity of
        [IEEE 1588-2008]"
    SYNTAX      Unsigned32 (0..65535)

PtpClockPortState ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This is the value of the current state of the protocol engine
        associated with this port."

    REFERENCE
        "Section 8.2.5.3.1 portState and 9.2.5 State machines of
        [IEEE 1588-2008]"
    SYNTAX          INTEGER  {
                        initializing(1),
                        faulty(2),
                        disabled(3),
                        listening(4),
                        preMaster(5),
                        master(6),
                        passive(7),
                        uncalibrated(8),
                        slave(9)
                    }

PtpClockPortTransportTypeAddress ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "255a"
    STATUS          current
    DESCRIPTION
```

```
        "The Clock port transport protocol address used for this
         communication between the clock nodes. This is a string
         corresponding to the address type as specified by the
         transport type used. The transport types can be defined
         elsewhere, in addition to the ones defined in this document.
         This can be an address of type IP version 4, IP version 6,
         Ethernet, DeviceNET, ControlNET or IEC61158. The OCTET STRING
         representation of the OID of ptpbaseWellKnownTransportTypes
         will be used in the values contained in the OCTET STRING."

    REFERENCE   "Annex D (IPv4), Annex E (IPv6), Annex F (Ethernet),
                 Annex G (DeviceNET), Annex H (ControlNET) and
                 Annex I (IEC61158) of [IEEE 1588-2008]"
    SYNTAX          OCTET STRING (SIZE (1..255))

PtpClockProfileType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "Clock Profile used. A profile is the set of allowed Precision
        Time Protocol (PTP) features applicable to a device."

    REFERENCE       "Section 3.1.30 profile and 19.3 PTP profiles of
                     [IEEE 1588-2008]"
    SYNTAX          INTEGER  {
                        default(1),
                        telecom(2),
                        vendorspecific(3)
                    }

PtpClockQualityAccuracyType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The ClockQuality as specified in sections 5.3.7, 7.6.2.5 and
        Table 6 of [IEEE 1588-2008].

        The following values are not represented in the enumerated
        values.

                 0x01-0x1F Reserved
                 0x32-0x7F Reserved

        It is important to note that section 7.1.1 of [RFC 2578] allows
        for gaps and enumerate values starting at zero when indicated by
        the protocol."

    REFERENCE
        "Section 5.3.7 ClockQuality, 7.6.2.5 clockAccuracy and Table 6
        clockAccuracy enumeration of [IEEE 1588-2008]"
    SYNTAX          INTEGER  {
```

```
                    -- reserved00(0:31), 0x00 to 0x1F
                       nanoSecond25(32),     -- 0x20
                       nanoSecond100(33),    -- 0x21
                       nanoSecond250(34),    -- 0x22
                       microSec1(35),        -- 0x23
                       microSec2dot5(36),    -- 0x24
                       microSec10(37),       -- 0x25
                       microSec25(38),       -- 0x26
                       microSec100(39),      -- 0x27
                       microSec250(40),      -- 0x28
                       milliSec1(41),        -- 0x29
                       milliSec2dot5(42),    -- 0x2A
                       milliSec10(43),       -- 0x2B
                       milliSec25(44),       -- 0x2C
                       milliSec100(45),      -- 0x2D
                       milliSec250(46),      -- 0x2E
                       second1(47),          -- 0x2F
                       second10(48),         -- 0x30
                       secondGreater10(49), -- 0x31
                       unknown(254)          -- 0xFE
                    -- reserved255(255),     0xFF
                   }

PtpClockQualityClassType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The ClockQuality as specified in section 5.3.7 ClockQuality,
        7.6.2.4 clockClass and Table 5 clockClass specifications of
        [IEEE 1588-2008]."

    REFERENCE       "Section 5.3.7, 7.6.2.4 and Table 5 of
                     [IEEE 1588-2008]."
    SYNTAX          INTEGER  {
                     -- reserved(0), 0x00
                     -- reserved(1:5), 0x01 to 0x05
                        clockclass6(6), -- 0x06
                        clockclass7(7), -- 0x07
                        -- reserved(8), 0x08
                        -- reserved(9:10), 0x09 to 0x0A
                        -- reserved(11:12), 0x0B, 0x0C
                        clockclass13(13), -- 0x0D
                        clockclass14(14), -- 0x0E
                        -- reserved(15:51), 0x0F to 0x33
                        clockclass52(52), -- 0x34
                        -- reserved(53:57), 0x35 to 0x39
                        clockclass58(58) -- 0x3A
                        -- reserved(59:67), 0x3B to 0x43
                        -- otherprofiles(68:122), 0x44 to 0x7A
                        -- reserved(123:127), 0x7B to 0x7F
```

```
                    -- reserved(128:132), 0x80 to 0x84
                }

PtpClockRoleType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The Clock Role. The protocol generates a Master Slave
        relationship among the clocks in the system.

        Clock Role      Value
        -------------------------
        Master clock     1
        Slave clock      2      "
    SYNTAX          INTEGER  {
                        master(1),
                        slave(2)
                    }

PtpClockStateType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The clock state returned by a PTP engine.

        Clock State      Value
        ------------------------
        Freerun state       1
        Holdover state      2
        Acquiring state     3
        Freq_locked state   4
        Phase_aligned state 5  "
    SYNTAX          INTEGER  {
                        freerun(1),
                        holdover(2),
                        acquiring(3),
                        frequencyLocked(4),
                        phaseAligned(5)
                    }

PtpClockTimeInterval ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "255a"
    STATUS          current
    DESCRIPTION
        "This textual convention corresponds to the TimeInterval
        structure indicated in section 5.3.2 of [IEEE 1588-2008].
        It will be presented in the form of a character array.
        Network byte order is assumed."

    REFERENCE
        "Section 5.3.2 TimeInterval and section 7.7.2.1 Timer interval
```

```
         specification of [IEEE 1588-2008]"
    SYNTAX          OCTET STRING (SIZE (1..255))

PtpClockTimeSourceType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The ClockQuality as specified in Sections 5.3.7, 7.6.2.6 and
        Table 7 of [IEEE 1588-2008].

        The following values are not represented in the enumerated
        values.

        0xF0-0xFE  For use by alternate PTP profiles
        0xFF       Reserved

        It is important to note that section 7.1.1 RFC 2578 allows for
        gaps and enumerate values to start with zero when indicated by
        the protocol."

    REFERENCE       "Section 5.3.7, 7.6.2.6 and Table 7 of
                     [IEEE 1588-2008]."
    SYNTAX          INTEGER  {
                        atomicClock(16), -- 0x10
                        gps(32), -- 0x20
                        terrestrialRadio(48), -- 0x22
                        ptp(64), -- 0x40
                        ntp(80), -- 0x50
                        handSet(96), -- 0x60
                        other(144), -- 0x90
                        internalOscillator(160) -- 0xA0
                    }

PtpClockTxModeType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "Transmission mode.

        Unicast:       Using unicast communication channel.
        Multicast:     Using Multicast communication channel.
        multicast-mix: Using multicast-unicast communication channel"
    SYNTAX          INTEGER  {
                        unicast(1),
                        multicast(2),
                        multicastmix(3)
                    }

PtpClockType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
```

```
        "The clock types as defined in the MIB module description."

    REFERENCE
        "Section 6.5.1 PTP device types of [IEEE 1588-2008]."
    SYNTAX          INTEGER  {
                        ordinaryClock(1),
                        boundaryClock(2),
                        transparentClock(3),
                        boundaryNode(4)
                    }

ptpbaseMIBNotifs  OBJECT IDENTIFIER
    ::= { ptpbaseMIB 0 }

ptpbaseMIBObjects  OBJECT IDENTIFIER
    ::= { ptpbaseMIB 1 }

ptpbaseMIBConformance  OBJECT IDENTIFIER
    ::= { ptpbaseMIB 2 }

ptpbaseMIBSystemInfo  OBJECT IDENTIFIER
    ::= { ptpbaseMIBObjects 1 }

ptpbaseMIBClockInfo  OBJECT IDENTIFIER
    ::= { ptpbaseMIBObjects 2 }


ptpbaseSystemTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseSystemEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of count information about the PTP system for all
        domains."
    ::= { ptpbaseMIBSystemInfo 1 }

ptpbaseSystemEntry OBJECT-TYPE
    SYNTAX          PtpbaseSystemEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing count information about a
        single domain. New row entries are added when the PTP clock for
        this domain is configured, while the unconfiguration of the PTP
        clock removes it."
    INDEX           {
                        ptpDomainIndex,
                        ptpInstanceIndex
                    }
```

```
    ::= { ptpbaseSystemTable 1 }

PtpbaseSystemEntry ::= SEQUENCE {
        ptpDomainIndex           PtpClockDomainType,
        ptpInstanceIndex         PtpClockInstanceType,
        ptpDomainClockPortsTotal Gauge32
}

ptpDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
        logical group of PTP devices. The Clock Domain is a logical
        group of clocks and devices that synchronize with each other
        using the PTP protocol.

        0           Default domain
        1           Alternate domain 1
        2           Alternate domain 2
        3           Alternate domain 3
        4 - 127     User-defined domains
        128 - 255   Reserved"
    ::= { ptpbaseSystemEntry 1 }

ptpInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the Clock for this
        domain."
    ::= { ptpbaseSystemEntry 2 }

ptpDomainClockPortsTotal OBJECT-TYPE
    SYNTAX          Gauge32
    UNITS           "ptp ports"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the total number of clock ports
        configured within a domain in the system."
    ::= { ptpbaseSystemEntry 3 }



ptpbaseSystemDomainTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseSystemDomainEntry
```

```
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP system for all clock modes
        -- ordinary, boundary or transparent."
    ::= { ptpbaseMIBSystemInfo 2 }

ptpbaseSystemDomainEntry OBJECT-TYPE
    SYNTAX          PtpbaseSystemDomainEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        clock mode for the PTP system. A row entry gets added when PTP
        clocks are configured on the node."
    INDEX           { ptpbaseSystemDomainClockTypeIndex }
    ::= { ptpbaseSystemDomainTable 1 }

PtpbaseSystemDomainEntry ::= SEQUENCE {
        ptpbaseSystemDomainClockTypeIndex PtpClockType,
        ptpbaseSystemDomainTotals         Unsigned32
}

ptpbaseSystemDomainClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseSystemDomainEntry 1 }

ptpbaseSystemDomainTotals OBJECT-TYPE
    SYNTAX          Unsigned32
    UNITS           "domains"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the total number of PTP domains for this
        particular clock type configured in this node."
    ::= { ptpbaseSystemDomainEntry 2 }



ptpbaseSystemProfile OBJECT-TYPE
    SYNTAX          PtpClockProfileType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
```

```
        "This object specifies the PTP Profile implemented on the
        system."
    REFERENCE       "Section 19.3 PTP profiles of [IEEE 1588-2008]"
    ::= { ptpbaseMIBSystemInfo 3 }

ptpbaseClockCurrentDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockCurrentDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP clock Current Datasets for
        all domains."
    ::= { ptpbaseMIBClockInfo 1 }

ptpbaseClockCurrentDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockCurrentDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP clock Current Datasets for a domain."
    REFERENCE
        "[IEEE 1588-2008] Section 8.2.2 currentDS data set member
        specifications of [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockCurrentDSDomainIndex,
                        ptpbaseClockCurrentDSClockTypeIndex,
                        ptpbaseClockCurrentDSInstanceIndex
                    }
    ::= { ptpbaseClockCurrentDSTable 1 }

PtpbaseClockCurrentDSEntry ::= SEQUENCE {
        ptpbaseClockCurrentDSDomainIndex      PtpClockDomainType,
        ptpbaseClockCurrentDSClockTypeIndex   PtpClockType,
        ptpbaseClockCurrentDSInstanceIndex    PtpClockInstanceType,
        ptpbaseClockCurrentDSStepsRemoved     Unsigned32,
        ptpbaseClockCurrentDSOffsetFromMaster PtpClockTimeInterval,
        ptpbaseClockCurrentDSMeanPathDelay    PtpClockTimeInterval
}

ptpbaseClockCurrentDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
logical
        group of PTP devices."
    ::= { ptpbaseClockCurrentDSEntry 1 }
```

```
ptpbaseClockCurrentDSClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockCurrentDSEntry 2 }

ptpbaseClockCurrentDSInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockCurrentDSEntry 3 }

ptpbaseClockCurrentDSStepsRemoved OBJECT-TYPE
    SYNTAX          Unsigned32
    UNITS           "Steps"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The current clock dataset StepsRemoved value.

        This object specifies the distance measured by the number of
        Boundary clocks between the local clock and the Foreign master
        as indicated in the stepsRemoved field of Announce messages."
    REFERENCE
        "Section 8.2.2.2 stepsRemoved of [IEEE 1588-2008]"
    ::= { ptpbaseClockCurrentDSEntry 4 }

ptpbaseClockCurrentDSOffsetFromMaster OBJECT-TYPE
    SYNTAX          PtpClockTimeInterval
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the current clock dataset ClockOffset
        value. The value of the computation of the offset in time
        between a slave and a master clock."
    REFERENCE
        "Section 8.2.2.3 currentDS.offsetFromMaster of [IEEE 1588-2008]"
    ::= { ptpbaseClockCurrentDSEntry 5 }

ptpbaseClockCurrentDSMeanPathDelay OBJECT-TYPE
    SYNTAX          PtpClockTimeInterval
```

```
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the current clock dataset
        MeanPathDelay value.

        The mean path delay between a pair of ports as measured by the
        delay request-response mechanism."
    REFERENCE
        "Section 8.2.2.4 currentDS.meanPathDelay of [IEEE 1588-2008]"
    ::= { ptpbaseClockCurrentDSEntry 6 }



ptpbaseClockParentDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockParentDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP clock Parent Datasets for
        all domains."
    ::= { ptpbaseMIBClockInfo 2 }

ptpbaseClockParentDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockParentDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP clock Parent Datasets for a domain."
    REFERENCE
        "Section 8.2.3 parentDS data set member specifications of
        [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockParentDSDomainIndex,
                        ptpbaseClockParentDSClockTypeIndex,
                        ptpbaseClockParentDSInstanceIndex
                    }
    ::= { ptpbaseClockParentDSTable 1 }

PtpbaseClockParentDSEntry ::= SEQUENCE {
    ptpbaseClockParentDSDomainIndex           PtpClockDomainType,
    ptpbaseClockParentDSClockTypeIndex        PtpClockType,
    ptpbaseClockParentDSInstanceIndex         PtpClockInstanceType,
    ptpbaseClockParentDSParentPortIdentity    OCTET STRING,
    ptpbaseClockParentDSParentStats           TruthValue,
    ptpbaseClockParentDSOffset                PtpClockIntervalBase2,
    ptpbaseClockParentDSClockPhChRate         Integer32,
```

```
    ptpbaseClockParentDSGMClockIdentity        PtpClockIdentity,
    ptpbaseClockParentDSGMClockPriority1       Unsigned32,
    ptpbaseClockParentDSGMClockPriority2       Unsigned32,
    ptpbaseClockParentDSGMClockQualityClass    PtpClockQualityClassType,
    ptpbaseClockParentDSGMClockQualityAccuracy
PtpClockQualityAccuracyType,
    ptpbaseClockParentDSGMClockQualityOffset   Unsigned32
}

ptpbaseClockParentDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
logical
        group of PTP devices."
    ::= { ptpbaseClockParentDSEntry 1 }

ptpbaseClockParentDSClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockParentDSEntry 2 }

ptpbaseClockParentDSInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockParentDSEntry 3 }

ptpbaseClockParentDSParentPortIdentity OBJECT-TYPE
    SYNTAX          OCTET STRING(SIZE(1..256))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the value of portIdentity of the port on
        the master that issues the Sync messages used in synchronizing
        this clock."
    REFERENCE
        "Section 8.2.3.2 parentDS.parentPortIdentity of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 4 }
```

```
ptpbaseClockParentDSParentStats OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Parent Dataset ParentStats value.

        This value indicates whether the values of ParentDSOffset
        and ParentDSClockPhChRate have been measured and are valid.
        A TRUE value shall indicate valid data."
    REFERENCE
        "Section 8.2.3.3 parentDS.parentStats of [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 5 }

ptpbaseClockParentDSOffset OBJECT-TYPE
    SYNTAX          PtpClockIntervalBase2 (-128..127)
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Parent Dataset
        ParentOffsetScaledLogVariance value.

        This value is the variance of the parent clock's phase as
        measured by the local clock."
    REFERENCE
        "Section 8.2.3.4
        parentDS.observedParentOffsetScaledLogVariance
        [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 6 }

ptpbaseClockParentDSClockPhChRate OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the clock's parent dataset
        ParentClockPhaseChangeRate value.

        This value is an estimate of the parent clock's phase change
        rate as measured by the slave clock."
    REFERENCE
        "Section 8.2.3.5
        parentDS.observedParentClockPhaseChangeRate of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 7 }

ptpbaseClockParentDSGMClockIdentity OBJECT-TYPE
    SYNTAX          PtpClockIdentity
```

```
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the parent dataset Grandmaster clock
        identity."
    REFERENCE
        "Section 8.2.3.6 parentDS.grandmasterIdentity of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 8 }

ptpbaseClockParentDSGMClockPriority1 OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the parent dataset Grandmaster clock
        priority1."
    REFERENCE
        "Section 8.2.3.8 parentDS.grandmasterPriority1 of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 9 }

ptpbaseClockParentDSGMClockPriority2 OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the parent dataset grandmaster clock
        priority2."
    REFERENCE
        "Section 8.2.3.9 parentDS.grandmasterPriority2 of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 10 }

ptpbaseClockParentDSGMClockQualityClass OBJECT-TYPE
    SYNTAX          PtpClockQualityClassType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the parent dataset grandmaster clock
        quality class."
    REFERENCE
        "Section 8.2.3.7 parentDS.grandmasterClockQuality of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 11 }

ptpbaseClockParentDSGMClockQualityAccuracy OBJECT-TYPE
    SYNTAX          PtpClockQualityAccuracyType
    MAX-ACCESS      read-only
```

```
    STATUS          current
    DESCRIPTION
        "This object specifies the parent dataset grandmaster clock
        quality accuracy."
    REFERENCE
        "Section 8.2.3.7 parentDS.grandmasterClockQuality of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 12 }

ptpbaseClockParentDSGMClockQualityOffset OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the parent dataset grandmaster clock
        quality offset."
    REFERENCE
        "Section 8.2.3.7 parentDS.grandmasterClockQuality of
         [IEEE 1588-2008]"
    ::= { ptpbaseClockParentDSEntry 13 }



ptpbaseClockDefaultDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockDefaultDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP clock Default Datasets for
        all domains."
    ::= { ptpbaseMIBClockInfo 3 }

ptpbaseClockDefaultDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockDefaultDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP clock Default Datasets for a domain."
    INDEX           {
                        ptpbaseClockDefaultDSDomainIndex,
                        ptpbaseClockDefaultDSClockTypeIndex,
                        ptpbaseClockDefaultDSInstanceIndex
                    }
    ::= { ptpbaseClockDefaultDSTable 1 }

PtpbaseClockDefaultDSEntry ::= SEQUENCE {
        ptpbaseClockDefaultDSDomainIndex     PtpClockDomainType,
        ptpbaseClockDefaultDSClockTypeIndex  PtpClockType,
```

```
        ptpbaseClockDefaultDSInstanceIndex   PtpClockInstanceType,
        ptpbaseClockDefaultDSTwoStepFlag     TruthValue,
        ptpbaseClockDefaultDSClockIdentity   PtpClockIdentity,
        ptpbaseClockDefaultDSPriority1       Unsigned32,
        ptpbaseClockDefaultDSPriority2       Unsigned32,
        ptpbaseClockDefaultDSSlaveOnly       TruthValue,
        ptpbaseClockDefaultDSQualityClass    PtpClockQualityClassType,
        ptpbaseClockDefaultDSQualityAccuracy
PtpClockQualityAccuracyType,
        ptpbaseClockDefaultDSQualityOffset   Integer32
}

ptpbaseClockDefaultDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
logical
        group of PTP devices."
    ::= { ptpbaseClockDefaultDSEntry 1 }

ptpbaseClockDefaultDSClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockDefaultDSEntry 2 }

ptpbaseClockDefaultDSInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockDefaultDSEntry 3 }

ptpbaseClockDefaultDSTwoStepFlag OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies whether the Two Step process is used."
    ::= { ptpbaseClockDefaultDSEntry 4 }

ptpbaseClockDefaultDSClockIdentity OBJECT-TYPE
```

```
    SYNTAX          PtpClockIdentity
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default Datasets clock identity."
    ::= { ptpbaseClockDefaultDSEntry 5 }

ptpbaseClockDefaultDSPriority1 OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default Datasets clock Priority1."
    ::= { ptpbaseClockDefaultDSEntry 6 }

ptpbaseClockDefaultDSPriority2 OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default Datasets clock Priority2."
    ::= { ptpbaseClockDefaultDSEntry 7 }

ptpbaseClockDefaultDSSlaveOnly OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Whether the SlaveOnly flag is set."
    ::= { ptpbaseClockDefaultDSEntry 8 }

ptpbaseClockDefaultDSQualityClass OBJECT-TYPE
    SYNTAX          PtpClockQualityClassType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default dataset Quality Class."
    ::= { ptpbaseClockDefaultDSEntry 9 }

ptpbaseClockDefaultDSQualityAccuracy OBJECT-TYPE
    SYNTAX          PtpClockQualityAccuracyType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default dataset Quality Accuracy."
    ::= { ptpbaseClockDefaultDSEntry 10 }

ptpbaseClockDefaultDSQualityOffset OBJECT-TYPE
    SYNTAX          Integer32
```

```
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default dataset Quality offset."
    ::= { ptpbaseClockDefaultDSEntry 11 }



ptpbaseClockRunningTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockRunningEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP clock Running Datasets for
        all domains."
    ::= { ptpbaseMIBClockInfo 4 }

ptpbaseClockRunningEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockRunningEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP clock running Datasets for a domain."
    INDEX           {
                        ptpbaseClockRunningDomainIndex,
                        ptpbaseClockRunningClockTypeIndex,
                        ptpbaseClockRunningInstanceIndex
                    }
    ::= { ptpbaseClockRunningTable 1 }

PtpbaseClockRunningEntry ::= SEQUENCE {
        ptpbaseClockRunningDomainIndex     PtpClockDomainType,
        ptpbaseClockRunningClockTypeIndex  PtpClockType,
        ptpbaseClockRunningInstanceIndex   PtpClockInstanceType,
        ptpbaseClockRunningState           PtpClockStateType,
        ptpbaseClockRunningPacketsSent     Counter64,
        ptpbaseClockRunningPacketsReceived Counter64
}

ptpbaseClockRunningDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
        Logical group of PTP devices."
    ::= { ptpbaseClockRunningEntry 1 }
```

```
ptpbaseClockRunningClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockRunningEntry 2 }

ptpbaseClockRunningInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockRunningEntry 3 }

ptpbaseClockRunningState OBJECT-TYPE
    SYNTAX          PtpClockStateType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Clock state returned by a PTP
engine."

    ::= { ptpbaseClockRunningEntry 4 }

ptpbaseClockRunningPacketsSent OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the total number of all unicast and
        multicast packets that have been sent out for this clock in this
        domain for this type. These counters are discontinuous."
    ::= { ptpbaseClockRunningEntry 5 }

ptpbaseClockRunningPacketsReceived OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the total number of all unicast and
        multicast packets that have been received for this clock in this
        domain for this type. These counters are discontinuous."
    ::= { ptpbaseClockRunningEntry 6 }
```

```
ptpbaseClockTimePropertiesDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockTimePropertiesDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP clock time properties
        datasets for all domains."
    ::= { ptpbaseMIBClockInfo 5 }

ptpbaseClockTimePropertiesDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockTimePropertiesDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP clock timeproperties Datasets for a domain."
    REFERENCE
        "Section 8.2.4 timePropertiesDS data set member specifications
        of [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockTimePropertiesDSDomainIndex,
                        ptpbaseClockTimePropertiesDSClockTypeIndex,
                        ptpbaseClockTimePropertiesDSInstanceIndex
                    }
    ::= { ptpbaseClockTimePropertiesDSTable 1 }

PtpbaseClockTimePropertiesDSEntry ::= SEQUENCE {
  ptpbaseClockTimePropertiesDSDomainIndex           PtpClockDomainType,
  ptpbaseClockTimePropertiesDSClockTypeIndex        PtpClockType,
  ptpbaseClockTimePropertiesDSInstanceIndex
PtpClockInstanceType,
  ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid TruthValue,
  ptpbaseClockTimePropertiesDSCurrentUTCOffset      Integer32,
  ptpbaseClockTimePropertiesDSLeap59                TruthValue,
  ptpbaseClockTimePropertiesDSLeap61                TruthValue,
  ptpbaseClockTimePropertiesDSTimeTraceable         TruthValue,
  ptpbaseClockTimePropertiesDSFreqTraceable         TruthValue,
  ptpbaseClockTimePropertiesDSPTPTimescale          TruthValue,
  ptpbaseClockTimePropertiesDSSource
PtpClockTimeSourceType
}

ptpbaseClockTimePropertiesDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
```

```
        "This object specifies the domain number used to create a
logical
        group of PTP devices."
    ::= { ptpbaseClockTimePropertiesDSEntry 1 }

ptpbaseClockTimePropertiesDSClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockTimePropertiesDSEntry 2 }

ptpbaseClockTimePropertiesDSInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockTimePropertiesDSEntry 3 }

ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the timeproperties dataset value of
        whether the current UTC offset is valid."
    REFERENCE
        "Section 8.2.4.2 timePropertiesDS.currentUtcOffset of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 4 }

ptpbaseClockTimePropertiesDSCurrentUTCOffset OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the timeproperties dataset value of
        the current UTC offset.

        In PTP systems whose epoch is the PTP epoch, the value of
        timePropertiesDS.currentUtcOffset is the offset
        between TAI and UTC; otherwise the value has no meaning. The
        value shall be in units of seconds."
    REFERENCE
        "Section 8.2.4.3 timePropertiesDS.currentUtcOffsetValid of
```

```
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 5 }

ptpbaseClockTimePropertiesDSLeap59 OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Leap59 value in the clock Current
        Dataset."
    REFERENCE
        "Section 8.2.4.4 timePropertiesDS.leap59 of [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 6 }

ptpbaseClockTimePropertiesDSLeap61 OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Leap61 value in the clock Current
        Dataset."
    REFERENCE
        "Section 8.2.4.5 timePropertiesDS.leap61 of [IEEE 1588-2008]"
 ::= { ptpbaseClockTimePropertiesDSEntry 7 }

ptpbaseClockTimePropertiesDSTimeTraceable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Time Traceable value in the clock
        Current Dataset."
    REFERENCE
        "Section 8.2.4.6 timePropertiesDS.timeTraceable of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 8 }

ptpbaseClockTimePropertiesDSFreqTraceable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Frequency Traceable value in the
        clock Current Dataset."
    REFERENCE
        "Section 8.2.4.7 timePropertiesDS.frequencyTraceable of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 9 }
```

```
ptpbaseClockTimePropertiesDSPTPTimescale OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP Timescale value in the clock
        Current Dataset."
    REFERENCE
        "Section 8.2.4.8 timePropertiesDS.ptpTimescale of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 10 }

ptpbaseClockTimePropertiesDSSource OBJECT-TYPE
    SYNTAX          PtpClockTimeSourceType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Timesource value in the clock Current
        Dataset."
    REFERENCE
        "Section 8.2.4.9 timePropertiesDS.timeSource of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 11 }



ptpbaseClockTransDefaultDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockTransDefaultDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the PTP Transparent clock Default
        Datasets for all domains."
    ::= { ptpbaseMIBClockInfo 6 }

ptpbaseClockTransDefaultDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockTransDefaultDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        PTP Transparent clock Default Datasets for a domain."
    REFERENCE
        "Section 8.3.2 transparentClockDefaultDS data set member
        specifications of [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockTransDefaultDSDomainIndex,
                        ptpbaseClockTransDefaultDSInstanceIndex
                    }
```

```
    ::= { ptpbaseClockTransDefaultDSTable 1 }

PtpbaseClockTransDefaultDSEntry ::= SEQUENCE {
        ptpbaseClockTransDefaultDSDomainIndex   PtpClockDomainType,
        ptpbaseClockTransDefaultDSInstanceIndex PtpClockInstanceType,
        ptpbaseClockTransDefaultDSClockIdentity PtpClockIdentity,
        ptpbaseClockTransDefaultDSNumOfPorts    Counter32,
        ptpbaseClockTransDefaultDSDelay         PtpClockMechanismType,
        ptpbaseClockTransDefaultDSPrimaryDomain PtpClockDomainType
}

ptpbaseClockTransDefaultDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
logical
        group of PTP devices."
    ::= { ptpbaseClockTransDefaultDSEntry 1 }

ptpbaseClockTransDefaultDSInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockTransDefaultDSEntry 2 }

ptpbaseClockTransDefaultDSClockIdentity OBJECT-TYPE
    SYNTAX          PtpClockIdentity
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the value of the clockIdentity attribute
        of the local clock."
    REFERENCE
       "Section 8.3.2.2.1 transparentClockDefaultDS.clockIdentity of
       [IEEE 1588-2008]"
    ::= { ptpbaseClockTransDefaultDSEntry 3 }

ptpbaseClockTransDefaultDSNumOfPorts OBJECT-TYPE
    SYNTAX          Counter32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the number of PTP ports of the device.
         These counters are discontinuous."
```

```
    REFERENCE
        "Section 8.3.2.2.2 transparentClockDefaultDS.numberPorts of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTransDefaultDSEntry 4 }

ptpbaseClockTransDefaultDSDelay OBJECT-TYPE
    SYNTAX          PtpClockMechanismType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object, if the transparent clock is an end-to-end
        transparent clock, has the value of E2E; if the
        transparent clock is a peer-to-peer transparent clock, the
        value
        shall be P2P."
    REFERENCE
        "Section 8.3.2.3.1 transparentClockDefaultDS.delayMechanism of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTransDefaultDSEntry 5 }

ptpbaseClockTransDefaultDSPrimaryDomain OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the value of the primary syntonization
        domain. The initialization value shall be 0."
    REFERENCE
        "Section 8.3.2.3.2 transparentClockDefaultDS.primaryDomain of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockTransDefaultDSEntry 6 }



ptpbaseClockPortTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockPortEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the clock ports for a particular
        domain."
    ::= { ptpbaseMIBClockInfo 7 }

ptpbaseClockPortEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
```

```
        clock port."
    INDEX           {
                        ptpbaseClockPortDomainIndex,
                        ptpbaseClockPortClockTypeIndex,
                        ptpbaseClockPortClockInstanceIndex,
                        ptpbaseClockPortTablePortNumberIndex
                    }
    ::= { ptpbaseClockPortTable 1 }

PtpbaseClockPortEntry ::= SEQUENCE {
        ptpbaseClockPortDomainIndex            PtpClockDomainType,
        ptpbaseClockPortClockTypeIndex         PtpClockType,
        ptpbaseClockPortClockInstanceIndex     PtpClockInstanceType,
        ptpbaseClockPortTablePortNumberIndex   PtpClockPortNumber,
        ptpbaseClockPortName                   DisplayString,
        ptpbaseClockPortRole                   PtpClockRoleType,
        ptpbaseClockPortSyncTwoStep            TruthValue,
        ptpbaseClockPortCurrentPeerAddressType AutonomousType,
        ptpbaseClockPortCurrentPeerAddress
PtpClockPortTransportTypeAddress,
        ptpbaseClockPortNumOfAssociatedPorts   Gauge32
}

ptpbaseClockPortDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
        logical group of PTP devices."
    ::= { ptpbaseClockPortEntry 1 }

ptpbaseClockPortClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockPortEntry 2 }

ptpbaseClockPortClockInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortEntry 3 }
```

```
ptpbaseClockPortTablePortNumberIndex OBJECT-TYPE
    SYNTAX          PtpClockPortNumber
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP Portnumber for this port."
    ::= { ptpbaseClockPortEntry 4 }

ptpbaseClockPortName OBJECT-TYPE
    SYNTAX          DisplayString (SIZE  (1..64))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP clock port name configured on the
        node."
    ::= { ptpbaseClockPortEntry 5 }

ptpbaseClockPortRole OBJECT-TYPE
    SYNTAX          PtpClockRoleType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object describes the current role (slave/master) of the
        port."
    ::= { ptpbaseClockPortEntry 6 }

ptpbaseClockPortSyncTwoStep OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies that two-step clock operation between
        the PTP master and slave device is enabled."
    ::= { ptpbaseClockPortEntry 7 }

ptpbaseClockPortCurrentPeerAddressType OBJECT-TYPE
    SYNTAX          AutonomousType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the current peer's network address type
         used for PTP communication."
    ::= { ptpbaseClockPortEntry 8 }

ptpbaseClockPortCurrentPeerAddress OBJECT-TYPE
    SYNTAX          PtpClockPortTransportTypeAddress
    MAX-ACCESS      read-only
    STATUS          current
```

```
    DESCRIPTION
        "This object specifies the current peer's network address used
        for PTP communication."
    ::= { ptpbaseClockPortEntry 9 }

ptpbaseClockPortNumOfAssociatedPorts OBJECT-TYPE
    SYNTAX          Gauge32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies -
        For a master port - the number of PTP slave sessions (peers)
        associated with this PTP port.
        For a slave port - the number of masters available to this slave
        port (might or might not be peered)."
    ::= { ptpbaseClockPortEntry 10 }



ptpbaseClockPortDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockPortDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the clock ports dataset for a
        particular domain."
    ::= { ptpbaseMIBClockInfo 8 }

ptpbaseClockPortDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing port dataset information for
        a single clock port."
    INDEX           {
                        ptpbaseClockPortDSDomainIndex,
                        ptpbaseClockPortDSClockTypeIndex,
                        ptpbaseClockPortDSClockInstanceIndex,
                        ptpbaseClockPortDSPortNumberIndex
                    }
    ::= { ptpbaseClockPortDSTable 1 }

PtpbaseClockPortDSEntry ::= SEQUENCE {
        ptpbaseClockPortDSDomainIndex          PtpClockDomainType,
        ptpbaseClockPortDSClockTypeIndex       PtpClockType,
        ptpbaseClockPortDSClockInstanceIndex   PtpClockInstanceType,
        ptpbaseClockPortDSPortNumberIndex      PtpClockPortNumber,
        ptpbaseClockPortDSName                 DisplayString,
```

```
        ptpbaseClockPortDSPortIdentity         OCTET STRING,
        ptpbaseClockPortDSlogAnnouncementInterval PtpClockIntervalBase2,
        ptpbaseClockPortDSAnnounceRctTimeout   Integer32,
        ptpbaseClockPortDSlogSyncInterval      PtpClockIntervalBase2,
        ptpbaseClockPortDSMinDelayReqInterval  Integer32,
        ptpbaseClockPortDSPeerDelayReqInterval Integer32,
        ptpbaseClockPortDSDelayMech            PtpClockMechanismType,
        ptpbaseClockPortDSPeerMeanPathDelay    PtpClockTimeInterval,
        ptpbaseClockPortDSGrantDuration        Unsigned32,
        ptpbaseClockPortDSPTPVersion           Unsigned32
}

ptpbaseClockPortDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
        logical group of PTP devices."
    ::= { ptpbaseClockPortDSEntry 1 }

ptpbaseClockPortDSClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockPortDSEntry 2 }

ptpbaseClockPortDSClockInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortDSEntry 3 }

ptpbaseClockPortDSPortNumberIndex OBJECT-TYPE
    SYNTAX          PtpClockPortNumber
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP portnumber associated with this
        PTP port."
    ::= { ptpbaseClockPortDSEntry 4 }

ptpbaseClockPortDSName OBJECT-TYPE
```

```
    SYNTAX          DisplayString (SIZE  (1..64))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP clock port dataset name."
    ::= { ptpbaseClockPortDSEntry 5 }

ptpbaseClockPortDSPortIdentity OBJECT-TYPE
    SYNTAX          OCTET STRING(SIZE(1..256))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP clock port Identity."
    ::= { ptpbaseClockPortDSEntry 6 }

ptpbaseClockPortDSlogAnnouncementInterval OBJECT-TYPE
    SYNTAX          PtpClockIntervalBase2
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Announce message transmission
        interval associated with this clock port."
    ::= { ptpbaseClockPortDSEntry 7 }

ptpbaseClockPortDSAnnounceRctTimeout OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Announce receipt timeout associated
        with this clock port."
    ::= { ptpbaseClockPortDSEntry 8 }

ptpbaseClockPortDSlogSyncInterval OBJECT-TYPE
    SYNTAX          PtpClockIntervalBase2
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Sync message transmission interval."
    ::= { ptpbaseClockPortDSEntry 9 }

ptpbaseClockPortDSMinDelayReqInterval OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Delay_Req message transmission
```

```
        interval."
    ::= { ptpbaseClockPortDSEntry 10 }

ptpbaseClockPortDSPeerDelayReqInterval OBJECT-TYPE
    SYNTAX          Integer32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Pdelay_Req message transmission
        interval."
    ::= { ptpbaseClockPortDSEntry 11 }

ptpbaseClockPortDSDelayMech OBJECT-TYPE
    SYNTAX          PtpClockMechanismType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the delay mechanism used. If the clock
        is an end-to-end clock, the value of the is e2e, else if the
        clock is a peer to-peer clock, the value shall be p2p."
    ::= { ptpbaseClockPortDSEntry 12 }

ptpbaseClockPortDSPeerMeanPathDelay OBJECT-TYPE
    SYNTAX          PtpClockTimeInterval
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the peer meanPathDelay."
    ::= { ptpbaseClockPortDSEntry 13 }

ptpbaseClockPortDSGrantDuration OBJECT-TYPE
    SYNTAX          Unsigned32
    UNITS           "seconds"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the grant duration allocated by the
        master."
    ::= { ptpbaseClockPortDSEntry 14 }

ptpbaseClockPortDSPTPVersion OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP version being used."
    ::= { ptpbaseClockPortDSEntry 15 }
```

```
ptpbaseClockPortRunningTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockPortRunningEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the clock ports running datasets for
        a particular domain."
    ::= { ptpbaseMIBClockInfo 9 }

ptpbaseClockPortRunningEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortRunningEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing running dataset information
        about a single clock port."
    INDEX           {
                        ptpbaseClockPortRunningDomainIndex,
                        ptpbaseClockPortRunningClockTypeIndex,
                        ptpbaseClockPortRunningClockInstanceIndex,
                        ptpbaseClockPortRunningPortNumberIndex
                    }
    ::= { ptpbaseClockPortRunningTable 1 }

PtpbaseClockPortRunningEntry ::= SEQUENCE {
        ptpbaseClockPortRunningDomainIndex        PtpClockDomainType,
        ptpbaseClockPortRunningClockTypeIndex     PtpClockType,
        ptpbaseClockPortRunningClockInstanceIndex PtpClockInstanceType,
        ptpbaseClockPortRunningPortNumberIndex    PtpClockPortNumber,
        ptpbaseClockPortRunningName               DisplayString,
        ptpbaseClockPortRunningState              PtpClockPortState,
        ptpbaseClockPortRunningRole               PtpClockRoleType,
        ptpbaseClockPortRunningInterfaceIndex     InterfaceIndexOrZero,
        ptpbaseClockPortRunningTransport          AutonomousType,
        ptpbaseClockPortRunningEncapsulationType  AutonomousType,
        ptpbaseClockPortRunningTxMode             PtpClockTxModeType,
        ptpbaseClockPortRunningRxMode             PtpClockTxModeType,
        ptpbaseClockPortRunningPacketsReceived    Counter64,
        ptpbaseClockPortRunningPacketsSent        Counter64
}

ptpbaseClockPortRunningDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
```

```
        logical group of PTP devices."
    ::= { ptpbaseClockPortRunningEntry 1 }

ptpbaseClockPortRunningClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the clock type as defined in the
        Textual convention description."
    ::= { ptpbaseClockPortRunningEntry 2 }

ptpbaseClockPortRunningClockInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortRunningEntry 3 }

ptpbaseClockPortRunningPortNumberIndex OBJECT-TYPE
    SYNTAX          PtpClockPortNumber
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP portnumber associated with this
        clock port."
    ::= { ptpbaseClockPortRunningEntry 4 }

ptpbaseClockPortRunningName OBJECT-TYPE
    SYNTAX          DisplayString (SIZE  (1..64))
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP clock port name."
    ::= { ptpbaseClockPortRunningEntry 5 }

ptpbaseClockPortRunningState OBJECT-TYPE
    SYNTAX          PtpClockPortState
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the port state returned by PTP engine.

        initializing
        faulty
        disabled
        listening
```

```
        preMaster
        master
        passive
        uncalibrated
        slave        "
    ::= { ptpbaseClockPortRunningEntry 6 }

ptpbaseClockPortRunningRole OBJECT-TYPE
    SYNTAX          PtpClockRoleType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Clock Role."
    ::= { ptpbaseClockPortRunningEntry 7 }

ptpbaseClockPortRunningInterfaceIndex OBJECT-TYPE
    SYNTAX          InterfaceIndexOrZero
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the interface on the node being used by
        the PTP Clock for PTP communication."
    ::= { ptpbaseClockPortRunningEntry 8 }

ptpbaseClockPortRunningTransport OBJECT-TYPE
    SYNTAX          AutonomousType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the transport protocol being used for PTP
        communication (the mapping used)."
    ::= { ptpbaseClockPortRunningEntry 9 }

ptpbaseClockPortRunningEncapsulationType OBJECT-TYPE
    SYNTAX          AutonomousType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the type of encapsulation if the
        interface is adding extra layers (e.g., VLAN, Pseudowire
        encapsulation...) for the PTP messages."
    ::= { ptpbaseClockPortRunningEntry 10 }

ptpbaseClockPortRunningTxMode OBJECT-TYPE
    SYNTAX          PtpClockTxModeType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the clock transmission mode as
```

```
        unicast:       Using unicast communication channel.
        multicast:     Using Multicast communication channel.
        multicast-mix: Using multicast-unicast communication channel"
    ::= { ptpbaseClockPortRunningEntry 11 }

ptpbaseClockPortRunningRxMode OBJECT-TYPE
    SYNTAX          PtpClockTxModeType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the clock receive mode as

        unicast:       Using unicast communication channel.
        multicast:     Using Multicast communication channel.
        multicast-mix: Using multicast-unicast communication channel"
    ::= { ptpbaseClockPortRunningEntry 12 }

ptpbaseClockPortRunningPacketsReceived OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the packets received on the clock port
        (cumulative). These counters are discontinuous."
    ::= { ptpbaseClockPortRunningEntry 13 }

ptpbaseClockPortRunningPacketsSent OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the packets sent on the clock port
        (cumulative). These counters are discontinuous."
    ::= { ptpbaseClockPortRunningEntry 14 }



ptpbaseClockPortTransDSTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockPortTransDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about the Transparent clock ports running
        dataset for a particular domain."
    ::= { ptpbaseMIBClockInfo 10 }
```

```
ptpbaseClockPortTransDSEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortTransDSEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing clock port Transparent
        dataset information about a single clock port"
    INDEX           {
                        ptpbaseClockPortTransDSDomainIndex,
                        ptpbaseClockPortTransDSInstanceIndex,
                        ptpbaseClockPortTransDSPortNumberIndex
                    }
    ::= { ptpbaseClockPortTransDSTable 1 }

PtpbaseClockPortTransDSEntry ::= SEQUENCE {
        ptpbaseClockPortTransDSDomainIndex        PtpClockDomainType,
        ptpbaseClockPortTransDSInstanceIndex      PtpClockInstanceType,
        ptpbaseClockPortTransDSPortNumberIndex    PtpClockPortNumber,
        ptpbaseClockPortTransDSPortIdentity       PtpClockIdentity,
        ptpbaseClockPortTransDSlogMinPdelayReqInt PtpClockIntervalBase2,
        ptpbaseClockPortTransDSFaultyFlag         TruthValue,
        ptpbaseClockPortTransDSPeerMeanPathDelay  PtpClockTimeInterval
}

ptpbaseClockPortTransDSDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create a
        Logical group of PTP devices."
    ::= { ptpbaseClockPortTransDSEntry 1 }

ptpbaseClockPortTransDSInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortTransDSEntry 2 }

ptpbaseClockPortTransDSPortNumberIndex OBJECT-TYPE
    SYNTAX          PtpClockPortNumber
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP port number associated with this
        port."
```

```
    REFERENCE       "Section 7.5.2 Port Identity of [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 3 }

ptpbaseClockPortTransDSPortIdentity OBJECT-TYPE
    SYNTAX          PtpClockIdentity
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the value of the PortIdentity
        attribute of the local port."
    REFERENCE
        "Section 8.3.3.2.1 transparentClockPortDS.portIdentity of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 4 }

ptpbaseClockPortTransDSlogMinPdelayReqInt OBJECT-TYPE
    SYNTAX          PtpClockIntervalBase2
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the value of the logarithm to the
        base 2 of the minPdelayReqInterval."
    REFERENCE
       "Section 8.3.3.3.1 transparentClockPortDS.logMinPdelayReqInterval
        of [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 5 }

ptpbaseClockPortTransDSFaultyFlag OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the value TRUE if the port is faulty
        and FALSE if the port is operating normally."
    REFERENCE
        "Section 8.3.3.3.2 transparentClockPortDS.faultyFlag of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 6 }

ptpbaseClockPortTransDSPeerMeanPathDelay OBJECT-TYPE
    SYNTAX          PtpClockTimeInterval
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies, if the delayMechanism used is P2P, the
        value of the estimate of the current one-way propagation delay,
        i.e., <meanPathDelay> on the link attached to this port,
        computed using the peer delay mechanism. If the value of the
```

```
        delayMechanism used is E2E, then the value will be zero."
    REFERENCE
        "Section 8.3.3.3.3 transparentClockPortDS.peerMeanPathDelay of
        [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 7 }



ptpbaseClockPortAssociateTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF PtpbaseClockPortAssociateEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Table of information about a given port's associated ports.

        For a master port: multiple slave ports that have established
                           sessions with the current master port.
        For a slave port:  the list of masters available for a given
                           slave port.

        Session information (packets, errors) to be displayed based on
        availability and scenario."
    ::= { ptpbaseMIBClockInfo 11 }


--
-- Well Known transport types for PTP communication.
--
ptpbaseWellKnownTransportTypes OBJECT IDENTIFIER ::= {
ptpbaseMIBClockInfo 12 }

ptpbaseTransportTypeIPversion4 OBJECT-IDENTITY
    STATUS current
    DESCRIPTION
        "IP version 4"
    ::= { ptpbaseWellKnownTransportTypes 1 }

ptpbaseTransportTypeIPversion6 OBJECT-IDENTITY
   STATUS current
     DESCRIPTION
        "IP version 6"
     ::= { ptpbaseWellKnownTransportTypes 2 }

ptpbaseTransportTypeEthernet OBJECT-IDENTITY
   STATUS current
    DESCRIPTION
        "Ethernet"
     ::= { ptpbaseWellKnownTransportTypes 3 }
```

```
ptpbaseTransportTypeDeviceNET OBJECT-IDENTITY
   STATUS current
    DESCRIPTION
        "Device NET"
     ::= { ptpbaseWellKnownTransportTypes 4 }

ptpbaseTransportTypeControlNET OBJECT-IDENTITY
   STATUS current
    DESCRIPTION
        "Control NET"
     ::= { ptpbaseWellKnownTransportTypes 5 }


ptpbaseTransportTypeIEC61158 OBJECT-IDENTITY
   STATUS current
    DESCRIPTION
        "IEC61158"
     ::= { ptpbaseWellKnownTransportTypes 6 }


--
-- Well Known encapsulation types for PTP communication.
--
ptpbaseWellKnownEncapsulationTypes OBJECT IDENTIFIER ::= {
ptpbaseMIBClockInfo 13 }

ptpbaseEncapsulationTypeEthernet OBJECT-IDENTITY
    STATUS current
    DESCRIPTION
        "Ethernet Encapsulation type."
    ::= { ptpbaseWellKnownEncapsulationTypes 1 }


ptpbaseEncapsulationTypeVLAN OBJECT-IDENTITY
    STATUS current
    DESCRIPTION
        "VLAN Encapsulation type."
    ::= { ptpbaseWellKnownEncapsulationTypes 2 }

ptpbaseEncapsulationTypeUDPIPLSP OBJECT-IDENTITY
   STATUS current
     DESCRIPTION
        "UDP/IP over MPLS Encapsulation type."
     ::= { ptpbaseWellKnownEncapsulationTypes 3 }

ptpbaseEncapsulationTypePWUDPIPLSP OBJECT-IDENTITY
   STATUS current
     DESCRIPTION
        "UDP/IP Pseudowire over MPLS Encapsulation type."
```

```
     ::= { ptpbaseWellKnownEncapsulationTypes 4 }

ptpbaseEncapsulationTypePWEthernetLSP OBJECT-IDENTITY
   STATUS current
     DESCRIPTION
        "Ethernet Pseudowire over MPLS Encapsulation type."
     ::= { ptpbaseWellKnownEncapsulationTypes 5 }



ptpbaseClockPortAssociateEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortAssociateEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing information about a single
        associated port for the given clockport."
    INDEX           {
                        ptpClockPortCurrentDomainIndex,
                        ptpClockPortCurrentClockTypeIndex,
                        ptpClockPortCurrentClockInstanceIndex,
                        ptpClockPortCurrentPortNumberIndex,
                        ptpbaseClockPortAssociatePortIndex
                    }
    ::= { ptpbaseClockPortAssociateTable 1 }

PtpbaseClockPortAssociateEntry ::= SEQUENCE {
        ptpClockPortCurrentDomainIndex           PtpClockDomainType,
        ptpClockPortCurrentClockTypeIndex        PtpClockType,
        ptpClockPortCurrentClockInstanceIndex    PtpClockInstanceType,
        ptpClockPortCurrentPortNumberIndex       PtpClockPortNumber,
        ptpbaseClockPortAssociatePortIndex       Unsigned32,
        ptpbaseClockPortAssociateAddressType     AutonomousType,
        ptpbaseClockPortAssociateAddress
PtpClockPortTransportTypeAddress,
        ptpbaseClockPortAssociatePacketsSent     Counter64,
        ptpbaseClockPortAssociatePacketsReceived Counter64,
        ptpbaseClockPortAssociateInErrors        Counter64,
        ptpbaseClockPortAssociateOutErrors       Counter64
}

ptpClockPortCurrentDomainIndex OBJECT-TYPE
    SYNTAX          PtpClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the given port's domain number."
    ::= { ptpbaseClockPortAssociateEntry 1 }
```

```
ptpClockPortCurrentClockTypeIndex OBJECT-TYPE
    SYNTAX          PtpClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the given port's clock type."
    ::= { ptpbaseClockPortAssociateEntry 2 }

ptpClockPortCurrentClockInstanceIndex OBJECT-TYPE
    SYNTAX          PtpClockInstanceType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the instance of the clock for this clock
        type in the given domain."
    ::= { ptpbaseClockPortAssociateEntry 3 }

ptpClockPortCurrentPortNumberIndex OBJECT-TYPE
    SYNTAX          PtpClockPortNumber
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the PTP Port Number for the given port."
    ::= { ptpbaseClockPortAssociateEntry 4 }

ptpbaseClockPortAssociatePortIndex OBJECT-TYPE
    SYNTAX          Unsigned32 (1..65535)
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the associated port's serial number in
        the current port's context."
    ::= { ptpbaseClockPortAssociateEntry 5 }

ptpbaseClockPortAssociateAddressType OBJECT-TYPE
    SYNTAX          AutonomousType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the peer port's network address type used
        for PTP communication. The OCTET STRING representation of the
        OID of ptpbaseWellKnownTransportTypes will be used in the values
        contained in the OCTET STRING."
    ::= { ptpbaseClockPortAssociateEntry 6 }

ptpbaseClockPortAssociateAddress OBJECT-TYPE
    SYNTAX          PtpClockPortTransportTypeAddress
    MAX-ACCESS      read-only
    STATUS          current
```

```
    DESCRIPTION
        "This object specifies the peer port's network address used for
        PTP communication."
    ::= { ptpbaseClockPortAssociateEntry 7 }

ptpbaseClockPortAssociatePacketsSent OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of packets sent to this peer port from the current
        port. These counters are discontinuous."
    ::= { ptpbaseClockPortAssociateEntry 8 }

ptpbaseClockPortAssociatePacketsReceived OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "The number of packets received from this peer port by the
        current port. These counters are discontinuous."
    ::= { ptpbaseClockPortAssociateEntry 9 }

ptpbaseClockPortAssociateInErrors OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the input errors associated with the
        peer port. These counters are discontinuous."
    ::= { ptpbaseClockPortAssociateEntry 10 }

ptpbaseClockPortAssociateOutErrors OBJECT-TYPE
    SYNTAX          Counter64
    UNITS           "packets"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the output errors associated with the
        peer port. These counters are discontinuous."
    ::= { ptpbaseClockPortAssociateEntry 11 }


-- Conformance Information Definition

ptpbaseMIBCompliances  OBJECT IDENTIFIER
```

```
    ::= { ptpbaseMIBConformance 1 }

ptpbaseMIBGroups  OBJECT IDENTIFIER
    ::= { ptpbaseMIBConformance 2 }


ptpbaseMIBCompliancesSystemInfo MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "Compliance statement for agents that provide read-only support
        for PTPBASE-MIB to provide system level information of clock
        devices.
        Such devices can only be monitored using this MIB module.

        The Module is implemented with support for read-only.  In other
        words, only monitoring is available by implementing this
        MODULE-COMPLIANCE."
    MODULE          -- this module
    MANDATORY-GROUPS { ptpbaseMIBSystemInfoGroup }
    ::= { ptpbaseMIBCompliances 1 }

ptpbaseMIBCompliancesClockInfo MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "Compliance statement for agents that provide read-only support
        for PTPBASE-MIB to provide clock related information.
        Such devices can only be monitored using this MIB module.

        The Module is implemented with support for read-only. In other
        words, only monitoring is available by implementing this
        MODULE-COMPLIANCE."
    MODULE          -- this module
    MANDATORY-GROUPS {
                        ptpbaseMIBClockCurrentDSGroup,
                        ptpbaseMIBClockParentDSGroup,
                        ptpbaseMIBClockDefaultDSGroup,
                        ptpbaseMIBClockRunningGroup,
                        ptpbaseMIBClockTimepropertiesGroup
                     }
    ::= { ptpbaseMIBCompliances 2 }

ptpbaseMIBCompliancesClockPortInfo MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "Compliance statement for agents that provide read-only support
        for PTPBASE-MIB to provide clock port related information.
        Such devices can only be monitored using this MIB module.

        The Module is implemented with support for read-only. In other
```

```
        words, only monitoring is available by implementing this
        MODULE-COMPLIANCE."
    MODULE          -- this module
    MANDATORY-GROUPS {
                        ptpbaseMIBClockPortGroup,
                        ptpbaseMIBClockPortDSGroup,
                        ptpbaseMIBClockPortRunningGroup,
                        ptpbaseMIBClockPortAssociateGroup
                    }
    ::= { ptpbaseMIBCompliances 3 }

ptpbaseMIBCompliancesTransparentClockInfo MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "Compliance statement for agents that provide read-only support
        for PTPBASE-MIB to provide Transparent clock related
        information.
        Such devices can only be monitored using this MIB module.

        The Module is implemented with support for read-only. In other
        words, only monitoring is available by implementing this
        MODULE-COMPLIANCE."
    MODULE          -- this module
    MANDATORY-GROUPS {
                        ptpbaseMIBClockTranparentDSGroup,
                        ptpbaseMIBClockPortTransDSGroup
                    }
    ::= { ptpbaseMIBCompliances 4 }

ptpbaseMIBSystemInfoGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseSystemDomainTotals,
                        ptpDomainClockPortsTotal,
                        ptpbaseSystemProfile
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing system-wide
        information"
    ::= { ptpbaseMIBGroups 1 }

ptpbaseMIBClockCurrentDSGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockCurrentDSStepsRemoved,
                        ptpbaseClockCurrentDSOffsetFromMaster,
                        ptpbaseClockCurrentDSMeanPathDelay
                    }
    STATUS          current
    DESCRIPTION
```

```
        "Group which aggregates objects describing PTP Current Dataset
        information"
    ::= { ptpbaseMIBGroups 2 }

ptpbaseMIBClockParentDSGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockParentDSParentPortIdentity,
                        ptpbaseClockParentDSParentStats,
                        ptpbaseClockParentDSOffset,
                        ptpbaseClockParentDSClockPhChRate,
                        ptpbaseClockParentDSGMClockIdentity,
                        ptpbaseClockParentDSGMClockPriority1,
                        ptpbaseClockParentDSGMClockPriority2,
                        ptpbaseClockParentDSGMClockQualityClass,
                        ptpbaseClockParentDSGMClockQualityAccuracy,
                        ptpbaseClockParentDSGMClockQualityOffset
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Parent Dataset
        information"
    ::= { ptpbaseMIBGroups 3 }

ptpbaseMIBClockDefaultDSGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockDefaultDSTwoStepFlag,
                        ptpbaseClockDefaultDSClockIdentity,
                        ptpbaseClockDefaultDSPriority1,
                        ptpbaseClockDefaultDSPriority2,
                        ptpbaseClockDefaultDSSlaveOnly,
                        ptpbaseClockDefaultDSQualityClass,
                        ptpbaseClockDefaultDSQualityAccuracy,
                        ptpbaseClockDefaultDSQualityOffset
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Default Dataset
        information"
    ::= { ptpbaseMIBGroups 4 }

ptpbaseMIBClockRunningGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockRunningState,
                        ptpbaseClockRunningPacketsSent,
                        ptpbaseClockRunningPacketsReceived
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP running state
```

```
        information"
    ::= { ptpbaseMIBGroups 5 }

ptpbaseMIBClockTimepropertiesGroup OBJECT-GROUP
    OBJECTS  {
                ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid,
                ptpbaseClockTimePropertiesDSCurrentUTCOffset,
                ptpbaseClockTimePropertiesDSLeap59,
                ptpbaseClockTimePropertiesDSLeap61,
                ptpbaseClockTimePropertiesDSTimeTraceable,
                ptpbaseClockTimePropertiesDSFreqTraceable,
                ptpbaseClockTimePropertiesDSPTPTimescale,
                ptpbaseClockTimePropertiesDSSource
             }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Time Properties
        information"
    ::= { ptpbaseMIBGroups 6 }

ptpbaseMIBClockTranparentDSGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockTransDefaultDSClockIdentity,
                        ptpbaseClockTransDefaultDSNumOfPorts,
                        ptpbaseClockTransDefaultDSDelay,
                        ptpbaseClockTransDefaultDSPrimaryDomain
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Transparent
        Dataset
        information"
    ::= { ptpbaseMIBGroups 7 }

ptpbaseMIBClockPortGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockPortName,
                        ptpbaseClockPortSyncTwoStep,
                        ptpbaseClockPortCurrentPeerAddress,
                        ptpbaseClockPortNumOfAssociatedPorts,
                        ptpbaseClockPortCurrentPeerAddressType,
                        ptpbaseClockPortRole
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing information for a
        given PTP Port."
    ::= { ptpbaseMIBGroups 8 }
```

```
ptpbaseMIBClockPortDSGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockPortDSName,
                        ptpbaseClockPortDSPortIdentity,
                        ptpbaseClockPortDSlogAnnouncementInterval,
                        ptpbaseClockPortDSAnnounceRctTimeout,
                        ptpbaseClockPortDSlogSyncInterval,
                        ptpbaseClockPortDSMinDelayReqInterval,
                        ptpbaseClockPortDSPeerDelayReqInterval,
                        ptpbaseClockPortDSDelayMech,
                        ptpbaseClockPortDSPeerMeanPathDelay,
                        ptpbaseClockPortDSGrantDuration,
                        ptpbaseClockPortDSPTPVersion
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP Port Dataset
        information"
    ::= { ptpbaseMIBGroups 9 }

ptpbaseMIBClockPortRunningGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockPortRunningName,
                        ptpbaseClockPortRunningState,
                        ptpbaseClockPortRunningRole,
                        ptpbaseClockPortRunningInterfaceIndex,
                        ptpbaseClockPortRunningTransport,
                        ptpbaseClockPortRunningEncapsulationType,
                        ptpbaseClockPortRunningTxMode,
                        ptpbaseClockPortRunningRxMode,
                        ptpbaseClockPortRunningPacketsReceived,
                        ptpbaseClockPortRunningPacketsSent
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP running interface
        information"
    ::= { ptpbaseMIBGroups 10 }

ptpbaseMIBClockPortTransDSGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockPortTransDSPortIdentity,
                        ptpbaseClockPortTransDSlogMinPdelayReqInt,
                        ptpbaseClockPortTransDSFaultyFlag,
                        ptpbaseClockPortTransDSPeerMeanPathDelay
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing PTP TransparentDS
```

```
        information"
    ::= { ptpbaseMIBGroups 11 }

ptpbaseMIBClockPortAssociateGroup OBJECT-GROUP
    OBJECTS         {
                        ptpbaseClockPortAssociatePacketsSent,
                        ptpbaseClockPortAssociatePacketsReceived,
                        ptpbaseClockPortAssociateAddress,
                        ptpbaseClockPortAssociateAddressType,
                        ptpbaseClockPortAssociateInErrors,
                        ptpbaseClockPortAssociateOutErrors
                    }
    STATUS          current
    DESCRIPTION
        "Group which aggregates objects describing information on peer
        PTP ports for a given PTP clock-port."
    ::= { ptpbaseMIBGroups 12 }


END
```

## 5. Security Considerations

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

The following objects all have a MAX-ACCESS of read-only:

ptpDomainClockPortsTotal,
ptpbaseSystemDomainTotals,
ptpbaseSystemProfile expose general information about the clock system.

ptpbaseClockRunningState,
ptpbaseClockRunningPacketsSent,
ptpbaseClockRunningPacketsReceived expose a clock’s current running status.

ptpbaseClockCurrentDSStepsRemoved,

ptpbaseClockCurrentDSOffsetFromMaster,
ptpbaseClockCurrentDSMeanPathDelay expose the values of a clock's current dataset (currentDS).

ptpbaseClockParentDSParentPortIdentity,
ptpbaseClockParentDSParentStats,
ptpbaseClockParentDSOffset,
ptpbaseClockParentDSClockPhChRate,
ptpbaseClockParentDSGMClockIdentity,
ptpbaseClockParentDSGMClockPriority1,
ptpbaseClockParentDSGMClockPriority2,
ptpbaseClockParentDSGMClockQualityClass,
ptpbaseClockParentDSGMClockQualityAccuracy,
ptpbaseClockParentDSGMClockQualityOffset expose the values of a clock's parent dataset (parentDS).

ptpbaseClockDefaultDSTwoStepFlag,
ptpbaseClockDefaultDSClockIdentity,
ptpbaseClockDefaultDSPriority1,
ptpbaseClockDefaultDSPriority2,
ptpbaseClockDefaultDSSlaveOnly,
ptpbaseClockDefaultDSQualityClass,
ptpbaseClockDefaultDSQualityAccuracy,
ptpbaseClockDefaultDSQualityOffset expose the values of a clock's default dataset (defaultDS).

ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid,
ptpbaseClockTimePropertiesDSCurrentUTCOffset,
ptpbaseClockTimePropertiesDSLeap59,
ptpbaseClockTimePropertiesDSLeap61,
ptpbaseClockTimePropertiesDSTimeTraceable,
ptpbaseClockTimePropertiesDSFreqTraceable,
ptpbaseClockTimePropertiesDSPTPTimescale,
ptpbaseClockTimePropertiesDSSource expose the values of a clock's time properties dataset (timePropertiesDS).

ptpbaseClockTransDefaultDSClockIdentity,
ptpbaseClockTransDefaultDSNumOfPorts,
ptpbaseClockTransDefaultDSDelay,
ptpbaseClockTransDefaultDSPrimaryDomain expose the values of a transparent clock's default dataset (transparentClockDefaultDS).

ptpbaseClockPortName,
ptpbaseClockPortRole,
ptpbaseClockPortSyncTwoStep,
ptpbaseClockPortCurrentPeerAddressType,
ptpbaseClockPortCurrentPeerAddress,
ptpbaseClockPortNumOfAssociatedPorts expose general information about a clock port.

ptpbaseClockPortRunningName,
ptpbaseClockPortRunningState,
ptpbaseClockPortRunningRole,
ptpbaseClockPortRunningInterfaceIndex,
ptpbaseClockPortRunningTransport,
ptpbaseClockPortRunningEncapsulationType,
ptpbaseClockPortRunningTxMode,
ptpbaseClockPortRunningRxMode,
ptpbaseClockPortRunningPacketsReceived,
ptpbaseClockPortRunningPacketsSent expose a clock port's current running status.

ptpbaseClockPortDSName,
ptpbaseClockPortDSPortIdentity,
ptpbaseClockPortDSlogAnnouncementInterval,
ptpbaseClockPortDSAnnounceRctTimeout,
ptpbaseClockPortDSlogSyncInterval,
ptpbaseClockPortDSMinDelayReqInterval,
ptpbaseClockPortDSPeerDelayReqInterval,
ptpbaseClockPortDSDelayMech,
ptpbaseClockPortDSPeerMeanPathDelay,
ptpbaseClockPortDSGrantDuration,
ptpbaseClockPortDSPTPVersion expose the values of a clock port's port dataset (portDS).

ptpbaseClockPortTransDSPortIdentity,
ptpbaseClockPortTransDSlogMinPdelayReqInt,
ptpbaseClockPortTransDSFaultyFlag,
ptpbaseClockPortTransDSPeerMeanPathDelay expose the values of a transparent clock port's port dataset (transparentClockPortDS).

ptpbaseClockPortAssociateAddressType,
ptpbaseClockPortAssociateAddress,
ptpbaseClockPortAssociatePacketsSent,
ptpbaseClockPortAssociatePacketsReceived,
ptpbaseClockPortAssociateInErrors,
ptpbaseClockPortAssociateOutErrors expose information about a clock port's peer node.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET (read) the objects in this MIB module.

Implementations SHOULD provide the security features described by the SNMPv3 framework (see [RFC 3410]), and implementations claiming compliance to the SNMPv3 standard MUST include full support for authentication and privacy via the User-based Security Model (USM) [RFC 3414] with the AES cipher algorithm [RFC 3826]. Implementations

MAY also provide support for the Transport Security Model (TSM) [RFC 5591] in combination with a secure transport such as SSH [RFC 5592] or TLS/DTLS [RFC 6353].

Further, deployment of SNMP versions prior to SNMPv3 is NOT recommended. Instead, it is recommended to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to those objects only to those principals (users) that have legitimate rights to access them.

## 6. IANA Considerations

The MIB module defined in this document uses the following IANA-assigned OBJECT IDENTIFIER value recorded in the SMI Numbers registry:

| Descriptor | OBJECT IDENTIFIER value |
|---|---|
| ptpbaseMIB | { mib-2 xxx } |

[NOTE for IANA: Please allocate an object identifier at http://www.iana.org/assignments/smi-numbers for object ptpbaseMIB.]

## 7. References

### 7.1. Normative References

[IEEE 1588-2008] "IEEE Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std. 1588(TM)-2008, 24 July 2008

### 7.2. Informative References

[RFC 1155] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", STD 16, RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990

[RFC 1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[RFC 1212] Rose, M., and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991

[RFC 1215] M. Rose, "A Convention for Defining Traps for use with the

SNMP", RFC 1215, Performance Systems International, March 1991

[RFC 1901] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.

[RFC 1906] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.

[RFC 2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 Harvard University, March 1997.

[RFC 2578] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC 2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC 2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC 3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet Standard Management Framework", RFC 3410 SNMP Research, Inc., Network Associates Laboratories, Ericsson, December 2002.

[RFC 3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, Enterasys Networks, BMC Software, Inc., Lucent Technologies, December 2002

[RFC 3412] Case, J., Harrington D., Presuhn R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, SNMP Research, Inc., Enterasys Networks, BMC Software, Inc., Lucent Technologies, December 2002.

[RFC 3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, Nortel Networks, Secure Computing Corporation, December 2002.

[RFC 3414] Blumenthal, U., and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, Lucent Technologies, December 2002.

[RFC 3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, Lucent Technologies, BMC Software, Inc., Cisco Systems, Inc., December 2002.

[RFC 3416] Presuhn, R. (Ed.), "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, BMC Software, Inc., December 2002.

[RFC 3417] Presuhn, R. (Ed.), "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, BMC Software, Inc., December 2002.

[RFC 3826] Blumenthal, U., Maino, F, and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model", RFC 3826, Lucent Technologies, Andiamo Systems, Inc., Cisco Systems, Inc., June 2004.

[RFC 5591] Harrington, D., and W. Hardraker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, Huawei Technologies (USA), Cobham Analytic Solutions, June 2009.

[RFC 5592] Harrington, D., Salowey, J., and W. Hardraker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP) ", RFC 5592, Huawei Technologies (USA), Cisco Systems, Cobham Analytic Solutions, June 2009.

[RFC 5905] David L. Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, University of Delaware, June 2010.

[RFC 6353] Hardraker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, SPARTA, Inc., July 2011.

[IEEE 802.3-2012] "IEEE Standard for Ethernet", IEEE Std. 802.3 - 2015, 3 September 2015

[G.8265.1] "Precision time protocol telecom profile for frequency synchronization", ITU-T Recommendation G.8265.1, July 2014.

## 8. Acknowledgements

Thanks to John Linton and Danny Lee for valuable comments, and to Bert Wijnen, Kevin Gross, Alan Luchuk, Chris Elliot, Brian Haberman and Dan Romascanu for their reviews of this MIB module.

## 9. Author’s Addresses

Vinay Shankarkumar
Cisco Systems,
7100-9 Kit Creek Road,
Research Triangle Park,
NC 27709,
USA.

Email: vinays@cisco.com

Laurent Montini,
Cisco Systems,
11, rue Camille Desmoulins,
92782 Issy-les-Moulineaux,
France.

Email: lmontini@cisco.com

Tim Frost,
Calnex Solutions Ltd.,
Oracle Campus,
Linlithgow,
EH49 7LR,
UK.

Email: tim.frost@calnexsol.com

Greg Dowd,
Microsemi Inc.,
3870 North First Street,
San Jose,
CA 95134,
USA.

Email: greg.dowd@microsemi.com