NTP Working Group D. Sibold
Internet-Draft K. Teichel
Intended status: Standards Track PTB
Expires: January 7, 2016 S. Roettger
Google Inc.
R. Housley
Vigil Security
July 06, 2015

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

draft-ietf-ntp-cms-for-nts-message-04

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

Copyright Notice

Table of Contents

1. Introduction

This document provides details 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. They are presented below. Note that the NTS Message Object that is at the core of each structure does not necessarily contain all the data needed for the particular message type, but may contain only that data which needs to be secured directly with cryptographic operations using the CMS. Specific information about what is included can be found in Section 3.

NTS-Plain: This archetype is used for actual time synchronization messages (explicitly, the following 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 at all. Figure 1 illustrates this structure.

```
+------------------------------------------------------+
|                                                      |
| NTS Message Object                                   |
|                                                      |
|                                                      |
+------------------------------------------------------+
```

NTS-Encrypted-and-Signed: 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 2 illustrates this structure.

```
      +---------------------------------------------------------+
      |                                                         |
      | ContentInfo                                             |
      |                                                         |
      | +-----------------------------------------------------+ |
      | |                                                     | |
      | | SignedData                                          | |
      | |                                                     | |
      | | +-------------------------------------------------+ | |
      | | |                                                 | | |
      | | | EnvelopedData                                   | | |
      | | |                                                 | | |
      | | | +---------------------------------------------+ | | |
      | | | |                                             | | | |
      | | | | NTS Message Object                          | | | |
      | | | |                                             | | | |
      | | | |                                             | | | |
      | | | +---------------------------------------------+ | | |
      | | |                                                 | | |
      | | +-------------------------------------------------+ | |
      | |                                                     | |
      | +-----------------------------------------------------+ |
      |                                                         |
      +---------------------------------------------------------+
```

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 3 illustrates this structure.

```
+---------------------------------------------------------+
|                                                         |
| ContentInfo                                             |
|                                                         |
| +-----------------------------------------------------+ |
| |                                                     | |
| | SignedData                                          | |
| |                                                     | |
| | +-------------------------------------------------+ | |
| | |                                                 | | |
| | | NTS Message Object                              | | |
| | |                                                 | | |
| | |                                                 | | |
| | +-------------------------------------------------+ | |
| |                                                     | |
| +-----------------------------------------------------+ |
|                                                         |
+---------------------------------------------------------+
```

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 3), with the only difference being that messages SHOULD NOT be digitally signed. This archetype employs the CMS structure merely in order to transport certificates.

2.1. Fields of the employed CMS Content Types

Overall, three CMS content types are used for NTS messages by the archetypes above. Explicitly, those content types are 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 archetypes except NTS-Plain. The fields of the ContentInfo content type are used as follows:

contentType -- indicates the type of the associated content. For all archetypes which use ContentInfo (these are NTS-Certified, NTS-Signed and NTS-Encrypted-and-Signed), 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 all archetypes using ContentInfo, 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-Encrypted-and-Signed 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-Encrypted-and-Signed 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-Encrypted-and-Signed 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-Encrypted-and-Signed 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, it MUST be absent.

2.1.3. EnvelopedData

The EnvelopedData content type is used only in the NTS-Encrypted-and-Signed 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 be 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 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. Implementation Notes: ASN.1 Structures and Use of the CMS

This section presents some hints about the structures of the NTS message objects for the different message types when one wishes to implement the security mechanisms.

3.1. 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))
```

The following ASN.1 coded data types are also necessary for other types.

```
KeyEncryptionAlgorithmIdentifiers ::=
    SET OF KeyEncryptionAlgorithmIdentifier

ContentEncryptionAlgorithmIdentifiers ::=
    SET OF ContentEncryptionAlgorithmIdentifier
```

3.2. Unicast Messages

3.2.1. Association Messages

3.2.1.1. Message Type: "client_assoc"

This message is structured according to the NTS-Plain archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ClientAssocData" and structured as follows:

```
ClientAssocData ::= SEQUENCE {
    nonce            NTSNonce,
    clientId         SubjectKeyIdentifier,
    digestAlgos      DigestAlgorithmIdentifiers,
    keyEncAlgos      KeyEncryptionAlgorithmIdentifiers,
    contentEncAlgos  ContentEncryptionAlgorithmIdentifiers
}
```

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

```
id-clientAssocData OBJECT IDENTIFIER ::=
    {nts(TBD) association(1) clientassocdata(1)}
```

3.2.1.2. Message Type: "server_assoc"

This message is structured according to the NTS-Signed archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ServerAssocData" and structured as follows:

```
ServerAssocData ::= SEQUENCE {
    nonce                 NTSNonce,
    clientId              SubjectKeyIdentifier,
    digestAlgos           DigestAlgorithmIdentifiers,
    choiceDigestAlgo      DigestAlgorithmIdentifier,
    keyEncAlgos           KeyEncryptionAlgorithmIdentifiers,
    choiceKeyEncAlgo      KeyEncryptionAlgorithmIdentifier,
    contentEncAlgos       ContentEncryptionAlgorithmIdentifiers
    choiceContentEncAlgo  ContentEncryptionAlgorithmIdentifier
}
```

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

```
id-serverAssocData OBJECT IDENTIFIER ::=
    {nts(TBD) association(1) serverassocdata(2)}
```

3.2.2. Cookie Messages

3.2.2.1. Message Type: "client_cook"

This message is structured according to the NTS-Certified archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ClientCookieData" and 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(TBD) association(1) clientcookiedata(3)}
```

3.2.2.2. Message Type: "server_cook"

This message is structured according to the "NTS-Encrypted-and-Signed" archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ServerCookieData" and structured as follows:

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

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

```
id-serverCookieData OBJECT IDENTIFIER ::=
    {nts(TBD) association(3) servercookiedata(4)}
```

3.2.3. Time Synchronization Messages

3.2.3.1. Message Type: "time_request"

This message is structured according to the "NTS-Plain" archetype.

This message type requires additional data to that which is included in the NTS message object, namely it requires regular time synchronization data, as an unsecured packet from a client to a server would contain. The NTS message object itself is an ASN.1 object of type "TimeRequestSecurityData", whose structure is as follows:

```
TimeRequestSecurityData ::=
SEQUENCE {
    nonce_t         NTSNonce,
    digestAlgo      DigestAlgorithmIdentifier,
    keyInputValue   OCTET STRING (SIZE(16))
}
```

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

```
id-timeRequestSecurityData OBJECT IDENTIFIER ::=
    {nts(TBD) time(2) timerequestsecuritydata(1)}
```

3.2.3.2. Message Type: "time_response"

This message is also structured according to "NTS-Plain".

It requires two items of data in addition to that which is transported in the NTS message object. Like "time_request", it requires regular time synchronization data. Furthermore, it requires the Message Authentication Code (MAC) to be generated over the whole rest of the packet (including the NTS message object) and transported in some way. The NTS message object itself is an ASN.1 object of type "TimeResponseSecurityData", with the following structure:

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

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

```
id-timeResponseSecurityData OBJECT IDENTIFIER ::=
    {nts(TBD) time(2) timeresponsesecuritydata(2)}
```

3.3. Broadcast Messages

3.3.1. Broadcast Parameter Messages

3.3.1.1. Message Type: "client_bpar"

This first broadcast message is structured according to the NTS-Plain archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "BroadcastParameterRequest" and structured as follows:

```
BroadcastParameterRequest ::=
SEQUENCE {
    nonce     NTSNonce,
    clientId  SubjectKeyIdentifier
}
```

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

```
id-broadcastParameterRequest OBJECT IDENTIFIER ::=
    {nts(TBD) association(1) broadcastparameterrequest(5)}
```

3.3.1.2. Message Type: "server_bpar"

This message is structured according to "NTS-Signed". There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "BroadcastParameterResponse" and structured as follows:

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

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

```
id-broadcastParameterResponse OBJECT IDENTIFIER ::=
    {nts(TBD) association(3) broadcastparameterresponse(6)}
```

3.3.2. Broadcast Time Synchronization Message

3.3.2.1. Message Type: "server_broad"

This message is structured according to the "NTS-Plain" archetype. It requires regular broadcast time synchronization data in addition to that which is carried in the NTS message object. Like "time_response", this message type also requires a MAC, generated over all other data, to be transported within the packet. The NTS message object itself is an ASN.1 object of type "BroadcastTime". It has the following structure:

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

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

```
id-broadcastTime OBJECT IDENTIFIER ::=
    {nts(TBD) time(1) broadcasttime(3)}
```

3.3.3. Broadcast Keycheck

3.3.3.1. Message Type: "client_keycheck"

This message is structured according to the "NTS-Plain" archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ClientKeyCheckSecurityData" and structured as follows:

```
ClientKeyCheckSecurityData ::=
SEQUENCE {
    nonce_k           NTSNonce,
    interval_number   INTEGER,
    digestAlgo        DigestAlgorithmIdentifier,
    keyInputValue     OCTET STRING (SIZE(16))
}
```

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

```
id-clientKeyCheckSecurityData OBJECT IDENTIFIER ::=
    {nts(TBD) time(1) clientkeychecksecuritydata(6)}
```

3.3.3.2. Message Type: "server_keycheck"

This message is also structured according to "NTS-Plain". It requires only a MAC, generated over the NTS message object, to be included in the packet in addition to what the NTS message object itself contains. The latter is an ASN.1 object of type "ServerKeyCheckSecurityData", which is structured as follows:

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

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

```
id-serverKeyCheckSecurityData OBJECT IDENTIFIER ::=
    {nts(TBD) time(1) serverkeychecksecuritydata(7)}
```

4. 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 }
```

5. IANA Considerations

IANA needs to assign an object identifier for the 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.

[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.

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-08 (work in progress), March 2015.

[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.

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

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

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

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

Stephen Roettger
Google Inc.

Email: stephen.roettger@googlemail.com

Russ Housley
Vigil Security
918 Spring Knoll Drive
Herndon, VA 20170

Email: housley@vigilsec.com

NTP Working Group T. Mizrahi
Internet Draft Marvell
Intended status: Standards Track D. Mayer
Updates: 5905 Network Time Foundation
Expires: February 2016 August 16, 2015

The Network Time Protocol Version 4 (NTPv4) Extension Fields
draft-ietf-ntp-extension-field-04.txt

Abstract

The Network Time Protocol Version 4 (NTPv4) defines the optional usage of extension fields. An extension field, defined in RFC5905, 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. This document updates RFC5905 by clarifying some points regarding NTP extension fields and their usage with Message Authentication Codes (MAC).

Status of this Memo

Copyright Notice

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, as defined in Section 7.5 of [RFC5905].

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 (Section 16 of [RFC5905]).

The only currently defined extension field is the one used by the AutoKey protocol [RFC5906]. This extension field is always followed by a MAC, and Section 10 of [RFC5906] specifies the parsing rules that allow a host to distinguish between an extension field and a MAC. However, a MAC is not mandatory after an extension field; an NTPv4 packet can include one or more extension fields without including a MAC (Section 7.5 of [RFC5905]).

This document updates [RFC5905] by clarifying some points regarding the usage of extension fields. Specifically, this document updates Section 7.5 of [RFC5905], clarifying the relationship between extension fields and MACs, and defining the behavior of a host that receives an unknown extension field.

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 [RFC5905]
MAC	Message Authentication Code

3. NTP Extension Fields - RFC 5905 Update

This document updates Section 7.5 of [RFC5905] as follows:

OLD:

7.5. NTP Extension Field Format

In NTPv4, one or more extension fields can be inserted after the header and before the MAC, if a MAC is present. If a MAC is not present, one or more extension fields can be inserted after the header, according to the following rules:

- 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.

Other than defining the field format, this document makes no use of the field contents. An extension field contains a request or response message in the format shown in Figure 14.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          Field Type           |            Length             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      .                                                               .
      .                            Value                              .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Padding (as needed)                     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Extension Field Format

All extension fields are zero-padded to a word (four octets) boundary. The Field Type field is specific to the defined function and is not elaborated here. While the minimum field length containing required fields is four words (16 octets), a maximum field length remains to be established.

The Length field is a 16-bit unsigned integer that indicates the length of the entire extension field in octets, including the Padding field.

NEW:

7.5. NTP Extension Field Format

In NTPv4, one or more extension fields can be inserted after the header and before the MAC, if a MAC is present.

Other than defining the field format, this document makes no use of the field contents. An extension field contains a request or response message in the format shown in Figure 14.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          Field Type           |            Length             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      .                                                               .
      .                            Value                              .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Padding (as needed)                     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Extension Field Format

All extension fields are zero-padded to a word (four octets) boundary.

The Field Type field is specific to the defined function and is not elaborated here. If a host receives an extension field with an unknown Field Type value, the host SHOULD ignore the extension field and MAY 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 inconsistent with the total length of the rest of the packet.

While the minimum field length containing required fields is four words (16 octets), the maximum field length cannot be longer than 65532 octets due to the maximum size of the length field.

The Length field is a 16-bit unsigned integer that indicates the length of the entire extension field in octets, including the Padding field.

7.5.1 Extension Fields and MACs

7.5.1.1 Extension Fields in the Presence of a MAC

An extension field can be used in an NTP packet that includes a MAC, for example, as defined in [RFC5906]. A specification that defines a new extension field MUST specify whether the extension field requires a MAC or not. If the extension field requires a MAC, the extension field specification MUST define 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.

7.5.1.2 Multiple Extension Fields with a MAC

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.

7.5.1.3 MAC in the absence of an Extension field

A MAC MUST NOT be 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.

7.5.1.4 Extension Fields in the Absence of a MAC

If a MAC is not present, one or more extension fields can be inserted after the header, according to the following rules:

- 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.

4. 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.

5. IANA Considerations

There are no new IANA considerations implied by this document.

6. Acknowledgments

The authors thank Dave Mills for his insightful comments.

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.

[RFC5905] 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

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

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 21, 2016

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

October 19, 2015

Network Time Security
draft-ietf-ntp-network-time-security-11

Abstract

This document describes Network Time Security (NTS), a collection of measures that enable secure time synchronization with time servers using protocols like the Network Time Protocol (NTP) or the 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

Copyright Notice

Table of Contents

1. Introduction

Time synchronization protocols are increasingly utilized to synchronize clocks in networked infrastructures. Successful attacks against the time synchronization protocol can seriously degrade the reliable performance of such infrastructures. Therefore, time synchronization protocols have to be secured if they are applied in environments that are prone to malicious attacks. This can be accomplished either by utilization of external security protocols, like IPsec or TLS, 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 which enable these and possibly other protocols to verify the authenticity of the time server/master and the integrity of the time synchronization protocol packets. The utilization of these measures for a given specific time synchronization protocol has to be described in a separate document.

[RFC7384] specifies that a security mechanism for timekeeping must be designed in such a way that it does not degrade the quality of the time transfer. This implies that for time keeping the increase in bandwidth and message latency caused by the security measures should be small. Also, NTP as well as PTP work via UDP and connections are stateless on the server/master side. Therefore, all security measures in this document are designed in such a way that they add little demand for bandwidth, that the necessary calculations can be executed in a fast manner, and that the measures do not require a server/master to keep state of a connection.

2. Terminology

2.1. Terms and Abbreviations

MITM	Man In The Middle
NTS	Network Time Security
TESLA	Timed Efficient Stream Loss-tolerant Authentication
MAC	Message Authentication Code
HMAC	Keyed-Hash Message Authentication Code

2.2. Common Terminology for PTP and NTP

This document refers to different time synchronization protocols, in particular to both the PTP and the NTP. Throughout the document the term "server" applies to both a PTP master and an NTP server. Accordingly, the term "client" applies to both a PTP slave and an NTP client.

3. Security Threats

The document "Security Requirements of Time Protocols in Packet Switched Networks" [RFC7384] contains a profound analysis of security threats and requirements for time synchronization protocols.

4. Objectives

The objectives of the NTS specification are as follows:

- Authenticity: NTS enables the client to authenticate its time server(s).

- 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.

- Authorization: NTS enables the client to verify its time server’s authorization. NTS optionally enables the server to verify the client’s authorization as well.

- Request-Response-Consistency: NTS enables a client to match an incoming response to a request it has sent. NTS also enables the client to deduce from the response whether its request to the server has arrived without alteration.

- Applicability to Protocols: NTS can be used to secure different time synchronization protocols, specifically at least NTP and PTP.

- Integration with Protocols: A client or server running an NTS-secured version of a time protocol does not negatively affect other participants who are running unsecured versions of that protocol.

- Server-Side Statelessness: All security measures of NTS work without creating the necessity for a server to keep state of a connection.

- Prevention of Amplification Attacks: All communication introduced by NTS offers protection against abuse for amplification denial-of-service attacks.

5. NTS Overview

NTS initially verifies the authenticity of the time server and exchanges a symmetric key, the so-called cookie, as well as a key input value (KIV). The KIV can be opaque for the client. After the cookie and the KIV are exchanged, the client then uses them to protect the authenticity and the integrity of subsequent unicast-type time synchronization packets. In order to do this, a Message Authentication Code (MAC) 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_<b> (HMAC(server seed, KIV)),
```

with the server seed as the key, where KIV is the client's key input value, and where the application of the function MSB_<b> returns only the b most significant bits. The server seed is a random value of bit length b that the server possesses, which has to remain secret. The cookie deterministically depends on KIV as long as the server seed stays the same. The server seed has to be refreshed periodically in order to provide key freshness as required in [RFC7384]. See Section 7 for details on seed refreshing.

Since the server does not keep a state of the client, it has to recalculate the cookie each time it receives a unicast time synchronization request from the client. To this end, the client has to attach its KIV to each request (see Section 6.1).

For broadcast-type messages, authenticity and integrity of the time synchronization packets are also ensured by a MAC, which is attached to the time synchronization packet by the sender. Verification of the broadcast-type 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 server securely shares the last element of the chain with all clients. The server splits time into intervals of uniform duration and assigns each key to an interval in reverse order. At each time interval, the server sends a 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 disclosure of the key in its associated disclosure interval occurs. In order to be able to verify the timeliness of the packets, the client has to be loosely time synchronized with the server. This has to be accomplished before broadcast associations can be used. For checking timeliness of packets, NTS uses another, more rigorous check in addition to just the clock lookup 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, and integrated into the communication flow of existing time synchronization protocols, see [I-D.ietf-ntp-cms-for-nts-message], 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.

6.1. 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 exchange can be repeatedly performed as often as the client desires and as long as the integrity of the server’s time responses is verified successfully.

6.1.1. Preconditions for the Unicast Time Synchronization Exchange

Before this message exchange is available, there are some requirements that the client and server need to meet:

- They MUST negotiate the hash algorithm for the MAC used in the time synchronization messages. Authenticity and integrity of the communication MUST be ensured.

- The client MUST know a key input value KIV. Authenticity and integrity of the communication MUST be ensured.

- Client and server MUST exchange the cookie (which depends on the KIV as described in section Section 5). Authenticity, confidentiality and integrity of the communication MUST be ensured.

One way of realising these requirements is to use the Association and Cookie Message Exchanges described in Appendix B.

6.1.2. Goals of the Unicast Time Synchronization Exchange

The unicast time synchronization exchange:

- exchanges (unicast) time synchronization data as specified by the appropriate time synchronization protocol,

- guarantees authenticity and integrity of the response to the client,

- guarantees request-response-consistency to the client.

6.1.3. Message Type: "time_request"

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

- the NTS message ID "time_request",

- the negotiated version number,

- a nonce,

- the negotiated hash algorithm H,

- the client's key input value (for which the client knows the associated cookie).

6.1.4. Message Type: "time_response"

This message is sent by the server after it has received a time_request message. Prior to this the server MUST recalculate the client's cookie by using the received key input value 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 nonce transmitted in time_request,

- a MAC (generated with the cookie as key) for verification of all of the above data.

6.1.5. Procedure Overview of the Unicast Time Synchronization Exchange

For a unicast time synchronization exchange, the following steps are performed:

1. 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.

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 6.1.4.

3. The client awaits a reply in the form of a time_response message. Upon receipt, it checks:

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

   * 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 send another cookie request (because the cookie might have changed due to a server seed refresh). If both checks are successful, the client SHOULD continue time synchronization.

```
            +------------------------+
            | o Re-generate cookie   |
            | o Assemble response    |
            | o Generate MAC         |
            +-----------+------------+
                        |
                      <-+->

    Server ------------------------------------------------>
                    /|      \
           time_    /        \  time_
           request /          \ response
                  /           \|
    Client ------------------------------------------------>

           <------ Unicast time ------>  <- Client-side ->
                  synchronization           validity
                     exchange               checks
```

Procedure for unicast time synchronization exchange.

6.2. Broadcast Time Synchronization Exchange

6.2.1. Preconditions for the Broadcast Time Synchronization Exchange

Before this message exchange is available, there are some requirements that the client and server need to meet:

- The client MUST receive all the information necessary to process broadcast time synchronization messages from the server. This includes
  - the one-way functions used for building the key chain,
  - 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 communication of the data listed above MUST guarantee authenticity of the server, as well as integrity and freshness of the broadcast parameters to the client.

6.2.2. Goals of the Broadcast Time Synchronization Exchange

The broadcast time synchronization exchange:

- transmits (broadcast) time synchronization data from the server to the client as specified by the appropriate time synchronization protocol,
- guarantees to the client that the received synchronization data has arrived in a timely manner as required by the TESLA protocol and is trustworthy enough to be stored for later checks,
- additionally guarantees authenticity of a certain broadcast synchronization message in the client’s storage.

6.2.3. 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 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.

6.2.4. Procedure Overview of Broadcast Time Synchronization Exchange

A broadcast time synchronization message exchange consists of the following steps:

1. The server follows the TESLA protocol by regularly sending server_broad messages as described in Section 6.2.3, adhering to its own disclosure schedule.

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

   * 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 loose time synchronization. If this is successful, the client obtains a stronger guarantee via a key check exchange (see below). 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.

   * 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 this check is successful, 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 it is falsified, the client MUST discard the packet.

   * If the disclosed key is legitimate, then the client verifies the authenticity of any packet that it has received during the corresponding time interval. If authenticity of a packet is verified, then it is released from the buffer and its time information can be utilized. If the verification fails, then authenticity is not given. In this case, the client MUST request authentic time from the server by means other than broadcast messages. Also, the client MUST re-initialize the broadcast sequence with a "client_bpar" message if the one-way key chain expires, which it can check via the disclosure schedule.

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

```
    Server ---------------------------------->
            \
             \  server_
              \ broad
              \|
    Client ---------------------------------->

            < Broadcast >  <- Client-side  ->
              time sync.      validity and
               exchange        timeliness
                                 checks
```

Procedure for broadcast time synchronization exchange.

6.3. Broadcast Keycheck

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

6.3.1. Preconditions for the Broadcast Keycheck Exchange

Before this message exchange is available, there are some requirements that the client and server need to meet:

- They MUST negotiate the hash algorithm for the MAC used in the time synchronization messages. Authenticity and integrity of the communication MUST be ensured.

- The client MUST know a key input value KIV. Authenticity and integrity of the communication MUST be ensured.

- Client and server MUST exchange the cookie (which depends on the KIV as described in section Section 5). Authenticity, confidentiality and integrity of the communication MUST be ensured.

These requirements conform to those for the unicast time synchronization exchange. Accordingly, they too can be realised via the Association and Cookie Message Exchanges described in Appendix B (Appendix B).

6.3.2. Goals of the Broadcast Keycheck Exchange

The keycheck exchange:

- guarantees to the client that the key belonging to the respective TESLA interval communicated in the exchange had not been disclosed before the client_keycheck message was sent.

- guarantees to the client the timeliness of any broadcast packet secured with this key if it arrived before client_keycheck was sent.

6.3.3. 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 NTS version number negotiated during association,
- a nonce,
- an interval number from the TESLA disclosure schedule,
- the hash algorithm H negotiated during association, and
- the client’s key input value KIV.

6.3.4. 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 received key input value and the transmitted hash algorithm. It contains

- the NTS message ID "server_keycheck"
- the version number as transmitted in "client_keycheck,
- the 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.

6.3.5. Procedure Overview of the Broadcast Keycheck Exchange

A broadcast keycheck message exchange consists of the following steps:

1. The client sends a client_keycheck message. It MUST memorize the nonce and the time interval number that it sends as a correlated pair.

2. Upon receipt of a client_keycheck message, the server looks up whether 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 6.3.4. For more details, see also Appendix C.

3. The client awaits a reply in the form of a server_keycheck message. On receipt, it performs the following checks:

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

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

   * that the TESLA interval number in that client_keycheck message matches the corresponding interval number from the server_keycheck, and

   * that the appended MAC verifies the received data.

```
                             +----------------------+
                             | o Assemble response  |
                             | o Re-generate cookie |
                             | o Generate MAC       |
                             +-----------+----------+
                                         |
                                       <-+->
    Server --------------------------------------------->
            \                        /|     \
             \  server_    client_   /       \  server_
              \ broad      keycheck /         \ keycheck
              \|                   /          \|
    Client --------------------------------------------->
             <-------- Extended broadcast time  ------->
                      synchronization exchange

                            <---- Keycheck exchange --->
```

Procedure for extended broadcast time synchronization exchange.

7. 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 8.1.

According to the requirements in [RFC7384], the server MUST refresh each server seed periodically. Consequently, 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 (Appendix B.4).

8. Hash Algorithms and MAC Generation

8.1. Hash Algorithms

Hash algorithms are used for calculation of the cookie and the MAC. The client and the server negotiate a hash algorithm H during the association phase at the beginning. The selected algorithm H is used for all hashing processes in that run.

In the TESLA scheme, 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 is non-negotiable.

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 its 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.

8.2. MAC Calculation

For the calculation of the MAC, client and server use a Keyed-Hash Message Authentication Code (HMAC) approach [RFC2104]. The HMAC is generated with the hash algorithm specified by the client (see Section 8.1).

9. IANA Considerations

10. Security Considerations

10.1. Privacy

The payload of time synchronization protocol packets of two-way time transfer approaches like NTP and PTP consists basically of time stamps, which are not considered secret [RFC7384]. Therefore, encryption of the time synchronization protocol packet’s payload is not considered in this document. However, an attacker can exploit the exchange of time synchronization protocol packets for topology detection and inference attacks as described in [RFC7624]. To make such attacks more difficult, that draft recommends the encryption of the packet payload. Yet, in the case of time synchronization protocols the confidentiality protection of time synchronization packet’s payload is of secondary importance since the packet’s meta data (IP addresses, port numbers, possibly packet size and regular sending intervals) carry more information than the payload. To enhance the privacy of the time synchronization partners, the usage of tunnel protocols such as IPsec and MACsec, where applicable, is therefore more suited than confidentiality protection of the payload.

10.2. Initial Verification of the Server Certificates

The client may wish to verify the validity of certificates during the initial association phase. Since it generally has no reliable time during this initial communication phase, it is impossible to verify the period of validity of the certificates. To solve this chicken-and-egg problem, the client has to rely on external means.

10.3. Revocation of Server Certificates

According to Section 7, 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 (Appendix B.3.3). Furthermore, certificates may also be revoked prior to the normal expiration date. To increase security the client MAY periodically verify the state of the server’s certificate via Online Certificate Status Protocol (OCSP) Online Certificate Status Protocol (OCSP) [RFC6960].

10.4. Mitigating Denial-of-Service for broadcast packets

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, the client and the 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 already knows, thus preventing memory flooding attacks.

Discussion: 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.

10.5. 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 from accurately measuring the network delay, and hence its time offset to the server [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 synchronization 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 utilize different paths for packet exchange as described in the I-D [I-D.ietf-tictoc-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 unicast-type messages: 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 for broadcast-type messages. 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 starts utilizing broadcast messages for time synchronization.

One possible way to achieve this initial synchronization is to establish a unicast association with its broadcast server until time synchronization and calibration of the packet delay time is achieved. After that, the client can establish a broadcast association with 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 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.

10.6. 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.

11. Acknowledgements

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

12. References

12.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[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>.

[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, DOI 10.17487/RFC4082, June 2005, <http://www.rfc-editor.org/info/rfc4082>.

[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>.

12.2. Informative References

[I-D.ietf-ntp-cms-for-nts-message]
Sibold, D., Teichel, K., Roettger, S., and R. Housley, "Protecting Network Time Security Messages with the Cryptographic Message Syntax (CMS)", draft-ietf-ntp-cms-for-nts-message-04 (work in progress), July 2015.

[I-D.ietf-tictoc-multi-path-synchronization]
Shpiner, A., Tse, R., Schelp, C., and T. Mizrahi, "Multi-Path Time Synchronization", draft-ietf-tictoc-multi-path-synchronization-02 (work in progress), April 2015.

[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.

[Mizrahi] Mizrahi, T., "A game theoretic analysis of delay attacks against time synchronization protocols", in Proceedings of Precision Clock Synchronization for Measurement Control and Communication, ISPCS 2012, pp. 1-6, September 2012.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[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>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <http://www.rfc-editor.org/info/rfc6960>.

[RFC7624] Barnes, R., Schneier, B., Jennings, C., Hardie, T., Trammell, B., Huitema, C., and D. Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement", RFC 7624, DOI 10.17487/RFC7624, August 2015, <http://www.rfc-editor.org/info/rfc7624>.

[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. (informative) TICTOC Security Requirements

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

Section	Requirement from RFC 7384	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	MAY	Optional,

	Authorization of Clients		Limited
5.2	Integrity protection	MUST	OK
5.3	Spoofing Prevention	MUST	OK
5.4	Protection from DoS attacks against the time protocol	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	SHOULD	OK
	Performance: bandwidth	SHOULD	OK
5.8	Confidentiality protection	MAY	NO
5.9	Protection against Packet Delay and Interception Attacks	MUST	Limited*)
5.10	Secure mode	MUST	OK
	Hybrid mode	SHOULD	-

*) See discussion in Section 10.5.

Comparison of NTS specification against Security Requirements of Time Protocols in Packet Switched Networks (RFC 7384)

Appendix B. (normative) Inherent Association Protocol Messages

This appendix presents a procedure that performs the association, the cookie, and also the broadcast parameter message exchanges between a client and a server. This procedure is one possible way to achieve the preconditions listed in Sections Section 6.1.1, Section 6.2.1, and Section 6.3.1 while taking into account the objectives given in Section Section 4.

B.1. Overview of NTS with Inherent Association Protocol

This inherent association protocol applies X.509 certificates to verify the authenticity of the time server and to exchange the cookie. This is done in two separate message exchanges, described below. An additional required exchange in advance serves to limit the amplification potential of the association message exchange.

A client needs a public/private key pair for encryption, with the public key enclosed in a certificate. A server needs a public/ private key pair for signing, with the public key enclosed in a certificate. If a participant intends to act as both a client and a server, it MUST have two different key pairs for these purposes.

If this protocol is employed, the hash value of the client’s certificate is used as the client’s key input value, i.e. the cookie is calculated according to:

```
cookie = MSB_<b> (HMAC(server seed, H(certificate of client))).
```

The client’s certificate contains the client’s public key and enables the server to identify the client, if client authorization is desired.

B.2. Access Message Exchange

This message exchange serves only to prevent the next (association) exchange from being abusable for amplification denial-of-service attacks.

B.2.1. Goals of the Access Message Exchange

The access message exchange:

- transfers a secret value from the server to the client (initiator),

- the secret value permits the client to initiate an association message exchange.

B.2.2. Message Type: "client_access"

This message is sent by a client who intends to perform an association exchange with the server in the future. It contains:

- the NTS message ID "client_access".

B.2.3. Message Type: "server_access"

This message is sent by the server on receipt of a client_access message. It contains:

- the NTS message ID "server_access",

- an access key.

B.2.4. Procedure Overview of the Access Exchange

For an access exchange, the following steps are performed:

1. The client sends a client_access message to the server.

2. Upon receipt of a client_access, the server calculates the access key according to

   access_key = HMAC(server seed; address of client),

   then it constructs and sends a reply in the form of a server_access message. In general the address of the client will be represented by the IP address of the client.

3. The client waits for a response in the form of a server_access message. Upon receipt of one, it MUST memorize the included access key.

B.3. Association Message Exchange

In this message exchange, the participants negotiate the hash and encryption algorithms that are used throughout the protocol. In addition, 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, the authenticity of future NTS messages from the server is ensured.

B.3.1. Goals of the Association Exchange

The association exchange:

- enables the client to verify any communication with the server as authentic,

- lets the participants negotiate NTS version and algorithms,

- guarantees authenticity and integrity of the negotiation result to the client,

- guarantees to the client that the negotiation result is based on the client’s original, unaltered request.

B.3.2. Message Type: "client_assoc"

This message is sent by the client if it wants to perform association with a server. It contains

- the NTS message ID "client_assoc",

- a nonce,

- the access key obtained earlier via an access message exchange,

- the version number of NTS that the client wants to use (this SHOULD be the highest version number that it supports),

- a selection of accepted hash algorithms, and

- a selection of accepted encryption algorithms.

B.3.3. 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 nonce transmitted in client_assoc,

- the client’s proposal for the version number, selection of accepted hash algorithms and selection of accepted encryption algorithms, as transmitted in client_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 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 that are included (see below), and

- a chain of certificates, which starts at the server and goes up to a trusted authority; each certificate MUST be certified by the one directly following it.

B.3.4. Procedure Overview of the Association Exchange

For an association exchange, the following steps are performed:

1. The client sends a client_assoc message to the server. It MUST keep the transmitted values for the version number and algorithms available for later checks.

2. Upon receipt of a client_assoc message, the server checks the validity of the included access key. If it is not valid, the server MUST abort communication. If it is valid, the server constructs and sends a reply in the form of a server_assoc message as described in Appendix B.3.3. Upon unsuccessful negotiation for version number or algorithms the server_assoc message MUST contain an error code.

3. The client 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 conforming version number.

   * It checks that the nonce sent back by the server matches the one transmitted in client_assoc,

   * It also verifies that the server has chosen the encryption and hash algorithms from its proposal sent in the client_assoc message and that this proposal was not altered.

- Furthermore, it performs authenticity checks on the certificate chain and the signature.

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

```
            +------------------------+
            | o Choose version       |
            | o Choose algorithms    |
            | o Acquire certificates |
            | o Assemble response    |
            | o Create signature     |
            +-----------+------------+
                        |
                      <-+->

    Server --------------------------->
                    /|     \
           client_  /       \ server_
           assoc   /         \ assoc
                  /          \|
    Client --------------------------->

           <------ Association ----->
                    exchange
```

Procedure for association and cookie exchange.

B.4. Cookie Message Exchange

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

B.4.1. Goals of the Cookie Exchange

The cookie exchange:

- enables the server to check the client’s authorization via its certificate (optional),
- supplies the client with the correct cookie and corresponding KIV for its association to the server,
- guarantees to the client that the cookie originates from the server and that it is based on the client’s original, unaltered request.

- guarantees that the received cookie is unknown to anyone but the server and the client.

B.4.2. 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",
- a nonce,
- the negotiated version number,
- the negotiated signature algorithm,
- the negotiated encryption algorithm,
- the negotiated hash algorithm H,
- the client’s certificate.

B.4.3. 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. 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.

B.4.4. Procedure Overview of the Cookie Exchange

For a cookie exchange, the following steps are performed:

1. The client sends a client_cook message to the server. The client MUST save the included nonce until the reply has been processed.

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. The server MAY use the client's certificate to check that the client is authorized to use the secure time synchronization service. With this, it MUST construct a server_cook message as described in Appendix B.4.3.

3. The client 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 beforehand.

   * 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 nonce and a cookie of the expected bit lengths.

   * 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.

```
        +----------------------------+
        | o OPTIONAL: Check client’s |
        |             authorization  |
        | o Generate cookie          |
        | o Encrypt inner message    |
        | o Generate signature       |
        +-------------+--------------+
                      |
                    <-+->

    Server ---------------------------->
                  /|     \
          client_ /       \ server_
          cook   /         \ cook
                /          \|
    Client ---------------------------->

           <--- Cookie exchange -->
```

Procedure for association and cookie exchange.

B.4.5. 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 successful association and cookie exchanges and only if it has made sure that its clock is roughly synchronized to the server’s.

See Appendix C for more details on TESLA.

B.4.5.1. Goals of the Broadcast Parameter Exchange

The broadcast parameter exchange

- provides the client with all the information necessary to process broadcast time synchronization messages from the server, and

- guarantees authenticity, integrity and freshness of the broadcast parameters to the client.

B.4.5.2. 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 NTS version number negotiated during association,

- a nonce, and

- the signature algorithm negotiated during association.

B.4.5.3. 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 nonce transmitted in client_bpar,

- 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.

B.4.5.4. Procedure Overview of the Broadcast Parameter Exchange

A broadcast parameter exchange consists of the following steps:

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

2. Upon receipt of a client_bpar message, the server constructs and sends a server_bpar message as described in Appendix B.4.5.3.

3. The client waits for a reply in the form of a server_bpar message, on which it performs the following checks:

- The message must contain all the necessary information for the TESLA protocol, as listed in Appendix B.4.5.3.

- The message must contain a nonce belonging to a client_bpar message that the client has previously sent.

- Verification of the message’s signature.

If any information is missing or if 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.

```
         +--------------------+
         | o Assemble response |
         | o Create public-key |
         |   signature         |
         +----------+----------+
                    |
                  <-+->

 Server --------------------------------------------->
                /|     \
        client_ /       \  server_
        bpar   /         \ bpar
              /          \|
 Client --------------------------------------------->

        <------- Broadcast ------>  <- Client-side ->
                 parameter              validity
                 exchange                checks
```

Procedure for unicast time synchronization exchange.

Appendix C. (normative) Using TESLA for Broadcast-Type Messages

For broadcast-type messages, 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 uses the "not re-using keys" scheme of TESLA as described in Section 3.7.2. of [RFC4082].

C.1. Server Preparation

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 shorter 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 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 its MAC has not yet been disclosed.

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

   It is RECOMMENDED that the server calculate d according to

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

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

```
   < - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                         Generation of Keys

             F              F               F                 F
    K_0  <-------- K_1  <--------  ...  <-------- K_{n-1} <------- K_n
     |              |                              |                |
     |              |                              |                |
     | F’           | F’                           | F’             | F’
     |              |                              |                |
     v              v                              v                v
    K’_0           K’_1            ...           K’_{n-1}         K’_n
             [______________|____       ____|_________________|_______]
                   I_1             ...            I_{n-1}          I_n

                     Course of Time/Usage of Keys
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ->
```

A schematic explanation of 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 at most one authenticated broadcast packet P_i. Such a 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 already received a packet with the same disclosed key. 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 = 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 it is 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 time stamp 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 client 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 the 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

$$T_i =< s_i, \text{ and}$$

$$s_i < T_{i+1}.$$

If it is falsified, the packet MUST be discarded and the client MUST reinitialize its broadcast module by performing time synchronization by other means than broadcast messages, and it MUST perform a new broadcast parameter exchange (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 the 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}. Upon success, it regards M_{i-d} as authenticated.

Appendix D. (informative) Dependencies

Issuer	Type	Owner	Description
Server PKI	private key (signature)	server	Used for server_assoc, server_cook, server_bpar. The server uses the private key to sign these messages. The client uses the public key to verify them. The certificate is used in server_assoc messages, for verifying authentication and (optionally) authorization.
	public key (signature)	client	
	certificate	server	
Client PKI	private key (encryption)	client	The server uses the client’s public key to encrypt the content of server_cook messages. The client uses the private key to decrypt them. The certificate is sent in client_cook messages, where it is used for transportation of the public key as well as (optionally) for verification of client authorization.
	public key (encryption)	server	
	certificate	client	

```
             +------------<---------------+
             |        At least one        |
             V        successful          |
      ++====[ ]===++               ++=====^=====++
      ||  Cookie  ||               ||Association||
      || Exchange ||               || Exchange  ||
      ++====_ _===++               ++===========++
             |
             |  At least one
             |  successful
             V
   ++=======[ ]=======++
   || Unicast Time    |>-----\   As long as further
   || Synchronization ||     |   synchronization
   || Exchange(s)     |<-----/   is desired
   ++=======_ _=======++
             |
              \                               Other (unspecified)
   Sufficient  \                          /   methods which give
   accuracy     \   either         or    /    sufficient accuracy
                 \----------\ /---------/
                             |
                             |
                             V
                  ++========[ ]=========++
                  || Broadcast          ||
                  || Parameter Exchange ||
                  ++========_ _=========++
                             |
                             |  One successful
                             |  per client
                             V
                   ++=======[ ]=======++
                   || Broadcast Time  |>--------\   As long as further
                   || Synchronization ||        |   synchronization
                   || Reception       |<--------/   is desired
                   ++=======_ _=======++
                             |
                            / \
                  either   /   \       or
               /----------/     \-------------\
              |                               |
              V                               V
   ++========[ ]========++         ++========[ ]========++
   || Keycheck Exchange ||         || Keycheck Exchange ||
   ++===================++         || with TimeSync     ||
                                   ++===================++
```

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

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2016

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

October 05, 2015

Using the Network Time Security Specification to Secure the Network Time Protocol
draft-ietf-ntp-using-nts-for-ntp-02

Abstract

This document describes how to use the measures described in the Network Time Security (NTS) specification to secure time synchronization with servers using the Network Time Protocol (NTP).

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

Copyright Notice

Table of Contents

1. Introduction

One of the most popular time synchronization protocols, the Network Time Protocol (NTP) [RFC5905], currently does not provide adequate intrinsic security precautions. The Network Time Security draft [I-D.ietf-ntp-network-time-security] specifies security measures which can be used to enable time synchronization protocols to verify authenticity of the time server and integrity of the time synchronization protocol packets.

This document provides detail on how to specifically use those measures to secure time synchronization between NTP clients and servers.

2. Objectives

The objectives of the NTS specification are as follows:

- Authenticity: NTS enables an NTP client to authenticate its time server(s).

- Integrity: NTS protects the integrity of NTP time synchronization protocol packets via a message authentication code (MAC).

- Confidentiality: NTS does not provide confidentiality protection of the time synchronization packets.

- Authorization: NTS optionally enables the server to verify the client’s authorization.

- Request-Response-Consistency: NTS enables a client to match an incoming response to a request it has sent. NTS also enables the client to deduce from the response whether its request to the server has arrived without alteration.

- Modes of operation: Both the unicast and the broadcast mode of NTP are supported.

- Hybrid mode: Both secure and insecure communication modes are possible for both NTP servers and clients.

- Compatibility:

    * Unsecured NTP associations are not affected.

    * An NTP server that does not support NTS is not affected by NTS-secured authentication requests.

3. Terms and Abbreviations

CMS Cryptographic Message Syntax [RFC5652]

HMAC Keyed-Hash Message Authentication Code

MAC Message Authentication Code

MITM Man In The Middle

NTP Network Time Protocol [RFC5905]

NTS Network Time Security

TESLA Timed Efficient Stream Loss-Tolerant Authentication [RFC4082]

4. Overview of NTS-Secured NTP

4.1. Symmetric and Client/Server Mode

The server does not keep a state of the client. NTS initially verifies the authenticity of the time server and exchanges a symmetric key, the so-called cookie and a key input value (KIV). The "association" and "cookie" message exchanges described in [I-D.ietf-ntp-network-time-security], Appendix B., can be utilized for the exchange of the cookie and KIV. An implementation MUST support the use of these exchanges. It MAY additionally support the use of any alternative secure communication for this purpose, as long as it fulfills the preconditions given in [I-D.ietf-ntp-network-time-security], Section 6.1.1.

After the cookie and KIV are exchanged, the participants then use them to protect the authenticity and the integrity of subsequent unicast-type time synchronization packets. In order to do this, the server attaches a Message Authentication Code (MAC) 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. Therefore, the client can perform a validity check for this MAC on reception of a time synchronization packet.

4.2. Broadcast Mode

After the client has accomplished the necessary initial time synchronization via client-server mode, the necessary broadcast parameters are communicated from the server to the client. The "broadcast parameter" message exchange described in [I-D.ietf-ntp-network-time-security], Appendix B., can be utilized for this communication. An implementation MUST support the use of this exchange. It MAY additionally support the use of any alternative secure communication for this purpose, as long as it fulfills the necessary security goals (given in [I-D.ietf-ntp-network-time-security], Section 6.2.1.).

After the client has received the necessry broadcast parameters, "broadcast time synchronization" message exchanges are utilized in combination with optional "broadcast keycheck" exchanges to protect authenticity and integrity of NTP broadcast time synchronization packets. As in the case of unicast time synchronization messages, this is also achieved by MACs.

5. Protocol Sequence

Throughout this section, the nonces, cookies and MACs mentioned have bit lengths of B_nonce, B_cookie and B_mac, respectively. These bit lengths are defined in Appendix B (Appendix B).

5.1. The Client

5.1.1. The Client in Unicast Mode

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

NOTE: Steps 1 through 4 MAY alternatively be replaced an alternative secure mechanism for association and cookie exchange. An implementation MAY choose to replace either steps 1 and 2 or all of the steps 1 through 4 by alternative secure communication.

Step 1: It sends a client_assoc message to the server. It MUST keep the transmitted nonce as well as the values for the version number and algorithms available for later checks.

Step 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 conforming version number.

* It checks that the nonce sent back by the server matches the one transmitted in client_assoc,

* It also verifies that the server has chosen the encryption and hash algorithms from its proposal sent in the client_assoc message and that this proposal was not altered.

- Furthermore, it performs authenticity checks on the certificate chain and the signature.

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 RFC 7384 [RFC7384] depends on the chosen trust anchor.

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

Step 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 beforehand.

- 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 B_nonce bit nonce and a B_cookie 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.

Step 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.

Step 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 previously,

- 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 3 (because the cookie might have changed due to a server seed refresh). If both checks are successful, the client SHOULD continue time synchronization by repeating the exchange of time_request and time_response messages.

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

5.1.2. 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 with it up to an upper bound for its time offset in unicast mode. After that, the client performs the following steps:

NOTE: Steps 1 and 2 MAY be replaced by an alternative security mechanism for the broadcast parameter exchange.

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

Step 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 specified for a server_bpar message.

- The message must contain a nonce belonging to a client_bpar message that the client has previously sent.

- Verification of the message’s signature.

If any information is missing or if 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.

Step 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 loose time synchronization. If this is successful, the client obtains 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 [I-D.ietf-ntp-network-time-security]. 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 it is 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 has 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. Also, the client MUST re-initialize the broadcast sequence with a "client_bpar" message if the one-way key chain expires, which it can check via the disclosure schedule.

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 ([I-D.ietf-ntp-network-time-security]) if the one-way key chain expires.

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

5.2. The Server

5.2.1. 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 [I-D.ietf-ntp-network-time-security].

- 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 [I-D.ietf-ntp-network-time-security]. With this, it MUST construct a server_cook message as described in [I-D.ietf-ntp-network-time-security].

- 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 [I-D.ietf-ntp-network-time-security].

The server MUST refresh its server seed periodically (see [I-D.ietf-ntp-network-time-security]).

In addition to the server MAY be ready to perform the following action:

- If an external mechanism for association and key exchange is used, the server has to react accordingly.

5.2.2. 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 [I-D.ietf-ntp-network-time-security].

- Upon receipt of a client_keycheck message, the server looks up whether 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 [I-D.ietf-ntp-network-time-security]. For more details, see also [I-D.ietf-ntp-network-time-security].

- The server follows the TESLA protocol in all other aspects, by regularly sending server_broad messages as described in [I-D.ietf-ntp-network-time-security], adhering to its own disclosure schedule.

The server is responsible to watch for the expiration date of the one-way key chain and generate a new key chain accordingly.

In addition to the items above, the server MAY be ready to perform the following action:

- Upon receipt of external communication for the purpose of broadcast parameter exchange, the server reacts according to the way the external communication is specified.

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

This section presents some hints about the structures of the communication packets for the different message types when one wishes to implement NTS for NTP. See document [I-D.ietf-ntp-cms-for-nts-message] for descriptions of the archetypes for CMS structures as well as for the ASN.1 structures that are referenced here.

All extension fields mentioned in the following list are notified by a field type value signalling content related to NTS version 1.0.

6.1. Unicast Messages

6.1.1. Association Messages

6.1.1.1. Message Type: "client_assoc"

This message is realized as an NTP packet with an extension field which holds an "NTS-Plain" archetype structure. This structure consists only of an NTS message object of the type "ClientAssocData", which holds all the data necessary for the NTS security mechanisms.

6.1.1.2. Message Type: "server_assoc"

Like "client_assoc", this message is realized as an NTP packet with an extension field which holds an "NTS-Plain" archetype structure, i.e. just an NTS message object of the type "ServerAssocData". The latter holds all the data necessary for NTS.

6.1.2. Cookie Messages

6.1.2.1. Message Type: "client_cook"

This message type is realized as an NTP packet with an extension field which holds a CMS structure of archetype "NTS-Certified", containing in its core an NTS message object of the type "ClientCookieData". The latter holds all the data necessary for the NTS security mechanisms.

6.1.2.2. Message Type: "server_cook"

This message type is realized as an NTP packet with an extension field containing a CMS structure of archetype "NTS-Encrypted-and-Signed". The NTS message object in that structure is a "ServerCookieData" object, which holds all data required by NTS for this message type.

6.1.3. Time Synchronization Messages

6.1.3.1. Message Type: "time_request"

This message type 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" (packed in a CMS structure of archetype "NTS-Plain").

6.1.3.2. Message Type: "time_response"

This message is also realized as an NTP packet with regular NTP time synchronization data. The packet also has an extension field which contains an ASN.1 object of type "TimeResponseSecurityData". Finally, this NTP packet has another extension field which contains a Message Authentication Code generated over the whole packet (including the extension field).

6.2. Broadcast Messages

6.2.1. Broadcast Parameter Messages

6.2.1.1. Message Type: "client_bpar"

This first broadcast message is realized as an NTP packet which is empty except for an extension field which contains an ASN.1 object of type "BroadcastParameterRequest" (packed in a CMS structure of archetype "CMS-Plain"). This is sufficient to transport all data specified by NTS.

6.2.1.2. Message Type: "server_bpar"

This message type is realized as an NTP packet whose extension field carries the necessary CMS structure (archetype: "NTS-Signed"). The NTS message object in this case is an ASN.1 object of type "BroadcastParameterResponse".

6.2.2. Broadcast Time Synchronization Message

6.2.2.1. Message Type: "server_broad"

This message’s 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" (packed in a CMS structure with archetype "NTS-Plain"). In addition to all this, this packet has another extension field which contains a Message Authentication Code generated over the whole packet (including the extension field).

6.2.3. Broadcast Keycheck

6.2.3.1. Message Type: "client_keycheck"

This message is realized as an NTP packet with an extension field, which transports a CMS structure of archetype "NTS-Plain", containing an ASN.1 object of type "ClientKeyCheckSecurityData".

6.2.3.2. Message Type: "server_keycheck"

This message is also realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ServerKeyCheckSecurityData" (packed in a CMS structure of archetype "NTS-Plain"). Additionally, this NTP packet has another extension field which contains a Message Authentication Code generated over the whole packet (including the extension field).

7. Security Considerations

7.1. Employing Alternative Means for Association and Cookie Exchange

If an implementation uses alternative means to perform association and cookie exchange, it MUST make sure that an adversary cannot abuse the server to obtain a cookie belonging to a chosen KIV.

7.2. Usage of NTP Pools

The certification-based authentication scheme described in [I-D.ietf-ntp-network-time-security] is not applicable to the concept of NTP pools. Therefore, NTS is unable to provide secure usage of NTP pools.

7.3. Server Seed Lifetime

tbd

7.4. Supported Hash Algorithms

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.

8. 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.

9. References

9.1. Normative References

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

[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, DOI 10.17487/RFC4082, June 2005, <http://www.rfc-editor.org/info/rfc4082>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[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>.

9.2. Informative References

[I-D.ietf-ntp-cms-for-nts-message] Sibold, D., Teichel, K., Roettger, S., and R. Housley, "Protecting Network Time Security Messages with the Cryptographic Message Syntax (CMS)", draft-ietf-ntp-cms-for-nts-message-04 (work in progress), July 2015.

[I-D.ietf-ntp-network-time-security] Sibold, D., Roettger, S., and K. Teichel, "Network Time Security", draft-ietf-ntp-network-time-security-09 (work in progress), July 2015.

[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. Flow Diagrams of Client Behaviour

```
                        +---------------------+
                        |Association Messages |
                        +----------+----------+
                                   |
   +------------------------------>o
   |                               |
   |                               v
   |                       +---------------+
   |                       |Cookie Messages|
   |                       +-------+-------+
   |                               |
   |                               o<------------------------------+
   |                               |                               |
   |                               v                               |
   |                     +-------------------+                     |
   |                     |Time Sync. Messages|                     |
   |                     +---------+---------+                     |
   |                               |                               |
   |                               v                               |
   |                            +-----+                            |
   |                            |Check|                            |
   |                            +--+--+                            |
   |                               |                               |
   |            /------------------+------------------\            |
   |           v                   v                   v           |
   |     .-----------.      .-------------.        .-------.       |
   |    ( MAC Failure )    ( Nonce Failure )      ( Success )      |
   |     ’-----+-----’      ’------+------’        ’---+---’       |
   |           |                   |                   |           |
   |           v                   v                   v           |
   |    +-------------+     +-------------+     +--------------+   |
   |    |Discard Data |     |Discard Data |     |Sync. Process |   |
   |    +------+------+     +------+------+     +------+-------+   |
   |           |                   |                   |           |
   |           |                   |                   v           |
   +-----------+                   +------------------>o-----------+
```

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

```
                            +-----------------------------+
                            |Broadcast Parameter Messages |
                            +--------------+--------------+
                                           |
                                           o<--------------------------+
                                           |                           |
                                           v                           |
                            +-----------------------------+            |
                            |Broadcast Time Sync. Message |            |
                            +--------------+--------------+            |
                                           |                           |
   +-------------------------------------->o                           |
   |                                       |                           |
   |                                       v                           |
   |                             +-------------------+                 |
   |                             |Key and Auth. Check|                 |
   |                             +---------+---------+                 |
   |                                       |                           |
   |                      /----------------*----------------\          |
   |                     v                                   v         |
   |                .---------.                         .---------.    |
   |               ( Verified  )                       ( Falsified )   |
   |                ’----+----’                         ’----+----’    |
   |                     |                                   |         |
   |                     v                                   v         |
   |              +-------------+                        +-------+     |
   |              |Store Message|                        |Discard|     |
   |              +------+------+                        +---+---+     |
   |                     |                                   |         |
   |                     v                                   +---------o
   |             +---------------+                                     |
   |             |Check Previous |                                     |
   |             +-------+-------+                                     |
   |                     |                                             |
   |            /--------*--------\                                    |
   |           v                   v                                   |
   |      .---------.         .---------.                              |
   |     ( Verified  )       ( Falsified )                             |
   |      ’----+----’         ’----+----’                              |
   |           |                   |                                   |
   |           v                   v                                   |
   |    +-------------+   +-----------------+                          |
   |    |Sync. Process|   |Discard Previous |                          |
   |    +------+------+   +--------+--------+                          |
   |           |                   |                                   |
   +-----------+                   +-----------------------------------+
```

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

Appendix B. Bit Lengths for Employed Primitives

Define the following bit lengths for nonces, cookies and MACs:

B_nonce = 128,

B_cookie = 128, and

B_mac = 128.

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.

Copyright Notice

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.

```
    +--------+     +-------+     +-------+     +-------+     +--------+
    |Switch, |     |       |     |       |     |       |     |Switch, |
    | Router |-----|  LER  |-----|  LSR  |-----|  LER  |-----| Router |
    |        |     |  OC   |     |  TC   |     |  OC   |     |        |
    +--------+     +-------+     +-------+     +-------+     +--------+
                   /                                 \
    +-------+     /                                   \     +-------+
    |  LER  |    /                                     \    |  LER  |
    | Master|---/                                       \---| Slave |
    | Clock |                                               | Clock |
    +-------+                                               +-------+
```

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.

```
    +--------+     +-------+     +-------+     +-------+     +--------+
    |Switch, |     |       |     |       |     |       |     |Switch, |
    | Router |-----|  LER  |-----|  LSR  |-----|  LER  |-----| Router |
    | OC/BC  |     |  BC   |     |  TC   |     |  BC   |     | OC/BC  |
    +--------+     +-------+     +-------+     +-------+     +--------+
```

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.

```
    +--------+     +-------+     +-------+     +-------+     +--------+
    |Switch, |     |       |     |       |     |       |     |Switch, |
    | Router |-----|  LER  |-----|  LSR  |-----|  LER  |-----| Router |
    |OC/TC/BC|     |  TC   |     |  TC   |     |  TC   |     |OC/TC/BC|
    +--------+     +-------+     +-------+     +-------+     +--------+
```

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.

```
 +--------+     +-------+     +-------+     +-------+     +--------+
 |Switch, |     |       |     |       |     |       |     |Switch, |
 | Router |-----|  LER  |-----|  LSR  |-----|  LER  |-----| Router |
 | OC/BC  |     |  BC   |     |  BC   |     |  BC   |     | OC/BC  |
 +--------+     +-------+     +-------+     +-------+     +--------+
```

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.

```
+----------------------+
|   Timing LSP Label   |
+----------------------+
|        IPv4/6        |
+----------------------+
|         UDP          |
+----------------------+
|   timing message     |
+----------------------+
```

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).

```
                  +-----------------+  +----------------+
                  |Timing LSP Label |  |Timing LSP Label|
                  +-----------------+  +----------------+
                  |    PW Label     |  |    PW Label    |
                  +-----------------+  +----------------+
                  |  Control Word   |  |      IP        |
                  +-----------------+  +----------------+
                  |    Ethernet     |  |      UDP       |
                  |     Header      |  +----------------+
                  +-----------------+  | Timing message |
                  |S-VLAN (Optional)|  |                |
                  +-----------------+  +----------------+
                  |C-VLAN (Optional)|        (B)
                  +-----------------+
                  | Timing message  |
                  |                 |
                  +-----------------+
                         (A)
```

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

- o Time-stamp field Type
  - * Correction Field, time-stamp
- o 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 A. Shpiner
Internet Draft Technion - Israel Institute of Technology
Intended status: Experimental R. Tse
Expires: April 2016 C. Schelp
PMC-Sierra
T. Mizrahi
Marvell
October 18, 2015

Multi-Path Time Synchronization
draft-ietf-tictoc-multi-path-synchronization-03.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. Slave Diversity is a recently introduced approach, where the master and slave clocks (also known as server and client) are connected through multiple network paths, and the slave combines the information received through all paths to obtain a higher clock accuracy compared to the conventional one-path approach. This document 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. The multi-path approach can significantly contribute to clock accuracy, security and fault tolerance. The Multi-Path Precision Time Protocol (MPPTP) and Multi-Path Network Time Protocol (MPNTP) define an additional layer that extends the existing PTP and NTP without the need to modify these protocols. MPPTP and MPNTP also allow backward compatibility with nodes that do not support the multi-path extension.

Status of this Memo

Copyright Notice

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.

```
                                  ____
                           ______/    \_____
                       ___/                 \____
                  ____/                          \
      ____       /           path 1              /           ___
     /    \     /    ________________________    \          /   \
    /Master\____\___/                        \____\________/Slave\
    \Clock /    /   \________ _______________/     \       \Clock/
     \____/     \                                  /        \__ /
                 \____       path 2             __/
                      \_______           ______/
                              \_________/
```

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), respectively. These extensions are defined at the network layer and do 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 extension 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 for each of the two multi-path protocols; a variant that requires both master and slave nodes to support the multi-path protocol, 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 Protocols over IP Networks

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 protocol is run only by the slave and the master is not aware of its usage. In the second variant, all clocks must support the multi-path protocol.

The dual-ended protocol provides higher path diversity by using multiple IP addresses at both ends, the master and slave, while the single-ended protocol only uses multiple addresses at the slave. On the other hand, the dual-ended protocol can only be deployed when both the master and the slave support this protocol. Dual-ended and single-ended protocols 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. 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. It is applicable if the load balancing rules in the network are known. 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.

The descriptions in this section refer to the end-to-end scheme of PTP, but are similarly applicable to the peer-to-peer scheme. The MPNTP protocol 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 protocols by nature require 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.

5.1. 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.

5.1.1. Single-Ended MPPTP Synchronization Message Exchange

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

o 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.

o 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.

5.1.2. Single-Ended MPNTP Synchronization Message Exchange

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

- 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.

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

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

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

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

- 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.

- The client combines the information from all paths.

5.2. 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.

5.2.1. Dual-Ended MPPTP Synchronization Message Exchange

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

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

- 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.

5.2.2. 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.

5.3. Using Traceroute for Path Discovery

The protocols presented above use 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.

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 the path of the timing packets in the forward and reverse direction are the same, path discovery should only be performed at the slave.

5.4. 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 5.1.1. and 5.2.1.

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 the interoperability of the protocol. Hence, this document does not define a specific method to be used by the slave.

7. Security Considerations

The security aspects of time synchronization protocols are discussed in detail in [TICTOCSEC]. The methods describe 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].

8. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

9. Acknowledgments

The authors gratefully acknowledge the useful comments provided by Peter Meyer and Doug Arnold, as well as other comments received from the TICTOC working group participants.

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

10. References

10.1. 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.

10.2. 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.

[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.

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

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

[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
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, 32000 Israel

Email: shalex@tx.technion.ac.il

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

Email: Richard.Tse@pmcs.com

Craig Schelp
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: craig.schelp@pmcs.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

TICTOC Working Group
Internet Draft
Intended status: Standards Track
Expires: August 4, 2015

Doug Arnold
Meinberg-USA
Heiko Gerstung
Meinberg

Feb 4, 2015

Enterprise Profile for the Precision Time Protocol With Mixed Multicast and Unicast Messages

draft-ietf-tictoc-ptp-enterprise-profile-05.txt

Status of this Memo

Copyright Notice

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.

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 [IEEE1588] needs to know the identity of the time sources in the network (the Master Clocks).

The so-called "Best Master Clock Algorithm" ([IEEE1588] Clause 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 like 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 up to 99% of the received multicast messages because they carried information for some other node. PTPv2 also introduced so-called "PTP profiles" ([IEEE1588] Clause 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. A number of 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 [IEEE1588] Clause 13.4), defining an optional Best Master Clock Algorithm and a few other ways. PTP has its own management protocol (defined in [IEEE1588] Clause 15.2) but allows a PTP profile specify an alternative management mechanism, for example SNMP.

2. Conventions used in this document

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

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 the Ethernet MAC address is used 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. Slaves, 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 [NTP].

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 define 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 500 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.

5. Network Technology

This PTP profile SHALL operate only in networks characterized by UDP [RFC768] over either IPv4 [RFC791] or IPv6 [RFC2460], as described by Annexes D and E in [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.

Similar to NTP, PTP makes the assumption 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, [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 [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, as long as 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. Unicast Discovery and Unicast Message Negotiation options MUST NOT be utilized.

8. Requirements for Master Clocks

Master clocks SHALL obey the standard Best Master Clock Algorithm from [IEEE1588]. PTP systems using this profile MAY support multiple simultaneous Grandmasters as long as 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. A clock with a port in the master state SHOULD indicate the maximum adjustment to its internal clock within one sync interval. The maximum phase adjustment is indicated in the Enterprise Profile announce TLV field for Maximum Phase Adjustment.

The Announce Messages SHALL include a TLV which indicates that the clock is operating in the Enterprise Profile. The TLV shall have the following structure:

TLV Type (Enumeration16): ORGANIZATION_EXTENSION value = 0003 hex

Length Field (UInteger16): value = 10. The number of TLV octets

Organization Unique Identifier (3 Octets): The Organization ID value for IETF assigned by IEEE = 00005Ehex

IETF Profile number (UInteger8): value = 1

Revision number (UInteger8): value = 1

Port Number (UInteger16): The Port Number of the port transmitting the TLV. The all-ones Port Number, with value FFFFhex, is used to indicate that the identified profile is applicable to all ports on the clock.

Maximum Absolute Phase Adjustment Value within one sync interval (UInteger16): value

Maximum Phase Adjustment Units (Enumeration8):
Value 0 = unknown
Value 1 = seconds
Value 3 = milliseconds
Value 6 = microseconds
Value 9 = nanoseconds
Value 12 = picoseconds
Value 15 = femtoseconds
All other values reserved for future use

Slaves can use the Maximum Phase Adjustment to determine if the clock is slewing to rapidly for the applications which are of interest. For example if the clock set by slave is used to measure time intervals then it might be desired that that the amount which the time changes during the intervals is limited.

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 [IEEE1588], section 11.2.

Since Announce messages are sent as multicast messages slaves can obtain the IP addresses of 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 SHALL NOT alter the Enterprise Profile TLV of an Announce message, or any other part of an Announce 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. Any PTP management message which is sent with the targetPortIdentity.clockIdentity set to all 1s (all clocks) MUST be sent as a multicast message. Management messages with any other value of for the Clock Identity is intended for a specific clock and 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. Clocks operating in the Enterprise Profile SHALL NOT use Unicast Message Negotiation to determine message rates. Slave clocks operating in the Enterprise Profile SHALL NOT use Unicast Discovery to establish connection to Master clocks. Grandmaster Clusters are NOT ALLOWED. The Alternate Master option is also forbidden. Clocks operating in the Enterprise Profile SHALL NOT use Alternate Timescales.

14. Interoperation with Other PTP Profiles

Clocks operating in the Enterprise Profile will not interoperate with clocks operating in the Power Profile [C37.238], because the Enterprise Profile requires the End to End Delay Measurement Mechanism and the Power Profile requires the Peer to Peer Delay Measurement Mechanism.

Clocks operating in the Enterprise Profile will not interoperate with clocks operating in the Telecom Profile for Frequency Synchronization[G8265.1], because the Enterprise Profile forbids Unicast Message Negotiation. This feature is part of the default manner of operation for the Telecom Profile for Frequency Synchronization.

Clocks operating in the Enterprise Profile will interoperate with clocks operating in the Default Profile described in [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 operates 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 Profile will 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. 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.

16. IANA Considerations

There are no IANA requirements in this specification.

17. References

17.1. Normative References

[IEEE1588] IEEE std. 1588-2008, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems." July, 2008.
[RFC768] Postel, J., "User Datagram Protocol," RFC 768, August, 980.

[RFC791] "Internet Protocol DARPA Internet Program Protocol Specification," RFC 791, September, 1981.

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

[RFC2460] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, December, 1998.

17.2. Informative References

[C37.238] IEEE std. C37.238-2011, "IEEE Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications," July 2011.

[G8265.1] ITU-T G.8265.1/Y.1365.1, "Precision time protocol telecom profile for frequency synchronization," October 2011.

[G8271] ITU-T G.8271/Y.1366, "Time and Phase Synchronization Aspects of Packet Networks" February, 2012.

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

18. Acknowledgments

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.

19. Authors’ Addresses

Doug Arnold
Meinberg USA
228 Windsor River Rd
Windsor, CA 95492
USA

Email: doug.arnold@meinberg-usa.com

Heiko Gerstung
Meinberg Funkuhren GmbH & Co. KG
Lange Wand 9
D-31812 Bad Pyrmont
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 24, 2015
March 25, 2015

Precision Time Protocol Version 2 (PTPv2) Management Information Base
draft-ietf-tictoc-ptp-mib-07.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, specified in IEEE Std. 1588(TM)-2008.

This memo specifies a MIB module in a manner that is both compliant to the SNMPv2 SMI, and semantically identical to the peer SNMPv1 definitions.

Table of Contents

1. Introduction

This memo defines a portion of the Management Information Base (MIB) 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 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 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 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.

1.1. Relationship to other Profiles and MIBs

This MIB 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 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 MIBs defined as part of the profile, and this MIB is not intended to replace those MIBs.

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 IEEE1588 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

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].

- 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].

- 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].

- 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 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.

3. Overview

The objects defined in this MIB are to be used when describing the Precision Time Protocol (PTPv2).

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    "201207230000Z"
    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 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 is to support 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 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 MIBs defined as part of the profile, and this MIB is not intended to replace those MIBs.

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]
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.

As defined in [IEEE 1588-2008]:

Accuracy:

The mean of the time or frequency error between the clock under test and a perfect reference clock, over an ensemble of measurements. Stability is a measure of how the mean varies with respect to variables such as time, temperature, and so on, while the precision is a measure of the deviation of the error from the mean.

Atomic process:

A process is atomic if the values of all inputs to the process are not permitted to change until all of the results of the process are instantiated, and the outputs of the process are not visible to other processes until the processing of each output is complete.

Boundary clock:
A clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain. It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.

Boundary node clock:
A clock that has multiple Precision Time Protocol(PTP) ports in a domain and maintains the timescale used in the domain. It differs from a boundary clock in that the clock roles can change.

Clock:
A node participating in the Precision Time Protocol (PTP) that is capable of providing a measurement of the passage of time since a defined epoch.

Domain:
A logical grouping of clocks that synchronize to each other using the protocol, but that are not necessarily synchronized to clocks in another domain.

End-to-end transparent clock:
A transparent clock that supports the use of the end-to-end delay measurement mechanism between slave clocks and the master clock. Each node must measure the residence time of PTP event messages and accumulate it in Correction Field.

Epoch:
The origin of a timescale.

Event:
An abstraction of the mechanism by which signals or conditions are generated and represented.

Foreign master:
An ordinary or boundary clock sending Announce messages to another clock that is not the current master recognized by the other clock.

Grandmaster clock:
Within a domain, a clock that is the ultimate source of time for clock synchronization using the protocol.

Holdover:
A clock previously synchronized/syntonized to another clock (normally a primary reference or a master clock) but now free-running based on its own internal oscillator, whose frequency is being adjusted using data acquired while it had been synchronized/syntonized to the other clock. It is said to be in holdover or in the holdover mode, as long as it is within its accuracy requirements.

Link:
A network segment between two Precision Time Protocol ports supporting the peer delay mechanism of this standard. The peer delay mechanism is designed to measure the propagation time over such a link.

Management node:
A device that configures and monitors clocks.

Master clock:
In the context of a single Precision Time Protocol communication path, a clock that is the source of time to which all other clocks on that path synchronize.

Message timestamp point:
A point within a Precision Time Protocol event message serving as a reference point in the message. A timestamp is defined by the instant a message timestamp point passes the reference plane of a clock.

Multicast communication:
A communication model in which each Precision Time Protocol message sent from any PTP port is capable of being received and processed by all PTP ports on the same PTP communication path.

Node:
A device that can issue or receive Precision Time Protocol communications on a network.

One-step clock:
A clock that provides time information using a single event message.

On-pass support:
Indicates that each node in the synchronization chain from master to slave can support IEEE-1588.

Ordinary clock:
A clock that has a single Precision Time Protocol port in a domain and maintains the timescale used in the domain. It may serve as a source of time, i.e., be a master clock, or may synchronize to another clock, i.e., be a slave clock.

Parent clock:
The master clock to which a clock is synchronized.

Peer-to-peer transparent clock:
A transparent clock that, in addition to providing Precision Time Protocol event transit time information, also provides corrections for the propagation delay of the link connected to the port receiving the PTP event message. In the presence of peer-to-peer transparent clocks, delay measurements between slave clocks and the master clock are performed using the peer-to-peer delay measurement mechanism.

Phase change rate:
The observed rate of change in the measured time with respect to the reference time. The phase change rate is equal to the fractional frequency offset between the measured frequency and the reference frequency.

PortNumber:
An index identifying a specific Precision Time Protocol port on a PTP node.

Primary reference:
A source of time and or frequency that is traceable to international standards.

Profile:
The set of allowed Precision Time Protocol features applicable to a device.

Precision Time Protocol communication:
Information used in the operation of the protocol, transmitted in a PTP message over a PTP communication path.

Precision Time Protocol communication path:
The signaling path portion of a particular network enabling direct communication among ordinary and boundary clocks.

Precision Time Protocol node:
PTP ordinary, boundary, or transparent clock or a device that generates or parses PTP messages.

Precision Time Protocol port:
A logical access point of a clock for PTP communications to the communications network.

Recognized standard time source:
A recognized standard time source is a source external to Precision Time Protocol that provides time and/or frequency as appropriate that is traceable to the international standards laboratories maintaining clocks that form the basis for the International Atomic Time and Universal Coordinated Time timescales. Examples of these are GPS, NTP, and NIST timeservers.

Requestor:
The port implementing the peer-to-peer delay mechanism that initiates the mechanism by sending a Pdelay_Req message.

Responder:
The port responding to the receipt of a Pdelay_Req message as part of the operation of the peer-to-peer delay mechanism.

Synchronized clocks:
Two clocks are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of a single event at an arbitrary time differ by no more than that uncertainty.

Syntonized clocks:
Two clocks are syntonized if the duration of the second is the same on both, which means the time as measured by each advances

at the same rate. They may or may not share the same epoch.

Timeout:
A mechanism for terminating requested activity that, at least from the requester’s perspective, does not complete within the specified time.

Timescale:
A linear measure of time from an epoch.

Traceability:
A property of the result of a measurement or the value of a standard whereby it can be related to stated references, usually national or international standards, through an unbroken chain of comparisons all having stated uncertainties.

Translation device:
A boundary clock or, in some cases, a transparent clock that translates the protocol messages between regions implementing different transport and messaging protocols, between different versions of [IEEE 1588-2008], or different PTP profiles.

Transparent clock:
A device that measures the time taken for a Precision Time Protocol event message to transit the device and provides this information to clocks receiving this PTP event message.

Two-step clock:
A clock that provides time information using the combination of an event message and a subsequent general message.

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

Data type mapping	Textual Convention	SYNTAX
5.3.2 TimeInterval	ClockTimeInterval	OCTET STRING(SIZE(1..255))
5.3.3 Timestamp	ClockTimestamp	OCTET STRING(SIZE(6))
5.3.4 ClockIdentity	ClockIdentity	OCTET STRING(SIZE(1..255))
5.3.5 PortIdentity	ClockPortNumber	INTEGER(1..65535)
5.3.7 ClockQuality	ClockQualityClassType	

Simple master-slave hierarchy, section 6.6.2.4 [IEEE 1588-2008]:

```
                  +---------------+
                  |   Ordinary    |
                  |   Clock -1    |
                  | (GrandMaster) |
                  +-------M-------+
                          |
                          1
                          |
          +---------------S-----------------+
          |            Boundary             |
          |            Clock -1             |
          +-----M---------------------M-----+
                |                     |
                2                     3
                |                     |
          +-----S-----+  +------------S-------------+
          | Ordinary  |  |         Boundary         |
          | Clock -2  |  |         Clock -2         |
          +-----------+  +-----M--------------M-----+
                               |              |
                               4              5
                               |              |
                         +-----S-----+  +-----S-----+
                         | Ordinary  |  | Ordinary  |
                         | Clock -3  |  | Clock -4  |
                         +-----------+  +-----------+

          Grandmaster

          Boundary Clock(0-N)   Ordinary Clocks(0-N)
          Ordinary Clocks(0-N)
```

```
        Relationship cardinality:
           PTP system 1 : N PTP Clocks
           PTP Clock  1 : 1 Domain
           PTP Clock  1 : N PTP Ports
           PTP Ports  N : M Physical Ports (interface in IF-MIB)
        Transparent clock diagram, section 6.7.1.3 of [IEEE 1588-2008]:

                         +------------------------------+
                         |     Boundary clock - 1       |
                         +------------------------------+
                           |                         |
                           |                         |
                   +-- A --+                         B
                   |                                 |
        +----------------------+                     |
        |  Ordinary clock      |                     |
        +----------------------+                     |
                                          +----------------------+
        +----------------------+          |      End-to-end      |
        |  Ordinary clock 1-1  |----------|  transparent clock-  |
        +----------------------+          |        1 - 1         |
                                          +----------------------+
                                                     |
                                                     C
                                                     |
                                          +----------------------+
        +----------------------+          |      End-to-end      |
        |  Ordinary clock 1-2  |----------|  transparent clock-  |
        +----------------------+          |        1 - 2         |
                                          +----------------------+

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

      -- revision log

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


-- Textual Conventions

ClockDomainType ::= 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. The following values
        define the valid domains.

                 Value       Definition
                 ---------   -------------------
                 0           Default domain
                 1           Alternate domain 1
                 2           Alternate domain 2
                 3           Alternate domain 3
                 4 - 127     User-defined domains
                 128 - 255   Reserved"

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

ClockIdentity ::= 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 ClockIdentity should be taken from the
        IEEE EUI-64 individual assigned numbers as indicated in
        Section 7.5.2.2.2 of [IEEE 1588-2008].

        The EUI-64 address is divided into the following fields:

             OUI bytes (0-2)
             Extension identifier bytes (3-7)

        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].

             Company EUI-48 = 0xACDE4823456716
             EUI-64 = ACDE48FFFE23456716

        It is important to note the IEEE Registration Authority has
        deprecated the use of MAC-48 in any new design."

    REFERENCE       "Section 7.5.2.2.1 of [IEEE 1588-2008]"
    SYNTAX          OCTET STRING (SIZE (1..255))
```

```
ClockInstanceType ::= 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)

ClockIntervalBase2 ::= 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].

        The mean time interval between successive messages shall be
        represented as the logarithm to the base 2 of this time
        interval measured in seconds on the local clock of the device
        sending the message. The values of these logarithmic attributes
        shall be selected from integers in the range -128 to 127 subject
        to further limits established in an applicable PTP profile."

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

ClockMechanismType ::= 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].

        Delay mechanism   Value(hex)  Specification
        ---------------   ----------  -------------
        E2E                  01       The port is configured to use the
                                      delay request-response mechanism.
        P2P                  02       The port is configured to use the
                                      peer delay mechanism.
        DISABLED             FE       The port does not implement the
                                      delay mechanism."
    REFERENCE   "Sections 8.2.5.4.4, 6.6.4, 7.4.2 of [IEEE 1588-2008]."
    SYNTAX      INTEGER {
                    e2e(1),
                    p2p(2),
                    disabled(254)
                    }

ClockPortNumber ::= 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 and 5.3.5 of [IEEE 1588-2008]"
    SYNTAX      Unsigned32 (0..65535)

ClockPortState ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This is the value of the current state of the protocol engine
        associated with this port.
        Port state      Value     Description
        -----------------------------------------------------------
        initializing      1       In this state a port initializes
                                  its data sets, hardware, and
                                  communication facilities.
        faulty            2       The fault state of the protocol.
        disabled          3       The port shall not place any
                                  messages on its communication path.
        listening         4       The port is waiting for the
                                  announceReceiptTimeout to expire or
                                  to receive an Announce message from
                                  a master.
        preMaster         5       The port shall behave in all respects
                                  as though it were in the MASTER state
                                  except that it shall not place any
                                  messages on its communication path
                                  except for Pdelay_Req, Pdelay_Resp,
                                  Pdelay_Resp_Follow_Up, signaling, or
                                  management messages.
        master            6       The port is behaving as a master port.
        passive           7       The port shall not place any messages
```

```
                                   on its communication path except for
                                   Pdelay_Req, Pdelay_Resp,
                                   Pdelay_Resp_Follow_Up, or signaling
                                   messages, or management messages that
                                   are a required response to another
                                   management message
        uncalibrated      8        The local port is preparing to
                                   synchronize to the master port.
        slave             9        The port is synchronizing to the
                                   selected master port."

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

ClockPortTransportTypeAddress ::= 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 address of type IP version 4, IP version 6,
         Ethernet, DeviceNET, ControlNET and IEC61158."

    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))

ClockProfileType ::= 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 and 19.3 PTP profiles of
                     [IEEE 1588-2008]"
    SYNTAX          INTEGER  {
                        default(1),
                        telecom(2),
                        vendorspecific(3)
                    }

ClockQualityAccuracyType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The ClockQuality as specified in section 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 RFC2578 allows for
        gaps and enumerate values to start with zero when indicated by
        the protocol."

    REFERENCE       "Section 5.3.7, 7.6.2.5 and Table 6 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
                    }

ClockQualityClassType ::= TEXTUAL-CONVENTION
    DISPLAY-HINT    "d"
    STATUS          current
    DESCRIPTION
        "The ClockQuality as specified in section 5.3.7, 7.6.2.4 and
        Table 5 of [IEEE 1588-2008].

        Value     Description
        --------  ------------------------------------------------
               0  Reserved to enable compatibility with future
                  versions.
             1-5  Reserved
               6  Shall designate a clock that is synchronized
                  to a primary reference time source.  The
                  timescale distributed shall be PTP.  A
                  clockClass 6 clock shall not be a slave to
                  another clock in the domain.
               7  Shall designate a clock that has previously
                  been designated as clockClass 6 but that has
                  lost the ability to synchronize to a primary
                  reference time source and is in holdover mode
                  and within holdover specifications. The
                  timescale distributed shall be PTP.  A
                  clockClass 7 clock shall not be a slave to
                  another clock in the domain.
               8  Reserved.
            9-10  Reserved to enable compatibility with future
                  versions.
           11-12  Reserved.
              13  Shall designate a clock that is synchronized
                  to an application-specific source of time.
                  The timescale distributed shall be ARB.  A
                  clockClass 13 clock shall not be a slave to
                  another clock in the domain.
              14  Shall designate a clock that has previously
                  been designated as clockClass 13 but that
                  has lost the ability to synchronize to an
                  application-specific source of time and is
                  in holdover mode and within holdover
                  specifications. The timescale distributed
                  shall be ARB.  A clockClass 14 clock shall
                  not be a slave to another clock in the domain.
           15-51  Reserved.
              52  Degradation alternative A for a clock of
                  clockClass 7 that is not within holdover
                  specification.  A clock of clockClass 52
                  shall not be a slave to another clock in
                  the domain.
           53-57  Reserved.
              58  Degradation alternative A for a clock of
                  clockClass 14 that is not within holdover
                  specification. A clock of clockClass 58 shall
                  not be a slave to another clock in the domain.
           59-67  Reserved.
          68-122  For use by alternate PTP profiles.
         123-127  Reserved.
         128-132  Reserved.
         133-170  For use by alternate PTP profiles.
         171-186  Reserved.

             187  Degradation alternative B for a clock of
                  clockClass 7 that is not within holdover
                  specification. A clock of clockClass 187 may
                  be a slave to another clock in the domain.
         188-192  Reserved.
             193  Degradation alternative B for a clock of
                  clockClass 14 that is not within holdover
                  specification. A clock of clockClass 193 may
                  be a slave to another clock in the domain.
         194-215  Reserved.
         216-232  For use by alternate PTP profiles.
         233-247  Reserved.
             248  Default. This clockClass shall be used if
                  none of the other clockClass definitions apply.
         249-250  Reserved.
             251  Reserved for version 1 compatibility; see Clause 18.
         252-254  Reserved.
             255  Shall be the clockClass of a slave-only clock; see
                  9.2.2."

    REFERENCE       "Section 5.3.7, 7.6.2.4 and Table 5 of
                     [IEEE 1588-2008]."
    SYNTAX          Unsigned32 (0..255)

ClockRoleType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The Clock Role. The protocol generates a Master Slave
        relationship among the clocks in the system.

        Clock Role      Value     Description
        --------------------------------------------------------------
        Master clock     1        A clock that is the source of
                                  time to which all other clocks on
                                  that path synchronize.

        Slave clock      2        A clock which synchronizes to
                                  another clock (master)."
    SYNTAX          INTEGER  {
                        master(1),
                        slave(2)
                    }

ClockStateType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The clock state returned by PTP engine.

        Clock State      Value   Description
        --------------------------------------------------------------
        Freerun state       1   Applies to a slave device that is not
                                locked to a master. This is the initial
                                state a slave starts out with when it
                                is not getting any PTP packets from the
                                master or because of some other input
                                error (erroneous packets, etc).
        Holdover state      2   In this state the slave device is
                                locked to a master but communication
                                with the master has been lost or the
                                timestamps in the ptp packets are
                                incorrect.  Since the slave was
                                locked to the master, it can run in this
                                state, with similar accuracy for some
                                time.  If communication with the master
                                is not restored for an extended period
                                (dependent on the clock implementation),
                                the device should move to the FREERUN
                                state.

        Acquiring state     3   The slave device is receiving packets
                                from a master and is trying to acquire
                                a lock.

        Freq_locked state   4   Slave device is locked to the Master
                                with respect to frequency, but not phase
                                aligned

        Phase_aligned state 5   Locked to the master with respect to
                                frequency and phase."
    SYNTAX          INTEGER  {
                        freerun(1),
                        holdover(2),
                        acquiring(3),
                        frequencyLocked(4),
                        phaseAligned(5)
                    }

ClockTimeInterval ::= 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.

        The TimeInterval type represents time intervals.

             struct TimeInterval
             {
                  Integer64 scaledNanoseconds;
             };

        The scaledNanoseconds member is the time interval expressed in
        units of nanoseconds and multiplied by 2**16.

        Positive or negative time intervals outside the maximum range
        of this data type shall be encoded as the largest positive and
        negative values of the data type, respectively.

        For example, 2.5 ns is expressed as 0000 0000 0002 8000 in
        Base16."

    REFERENCE
        "Section 5.3.2 and setion 7.7.2.1 Timer interval
         specification of [IEEE 1588-2008]"
    SYNTAX          OCTET STRING (SIZE (1..255))

ClockTimeSourceType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The ClockQuality as specified in section 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 RFC2578 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
                    }

ClockTxModeType ::= 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)
                    }

ClockType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The clock types as defined in the MIB module description."

    REFERENCE       "Section 6.5.1 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           ClockDomainType,
        ptpInstanceIndex         ClockInstanceType,
        ptpDomainClockPortsTotal Gauge32
}

ptpDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create 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          ClockInstanceType
    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 router."
    INDEX           { ptpbaseSystemDomainClockTypeIndex }
    ::= { ptpbaseSystemDomainTable 1 }

PtpbaseSystemDomainEntry ::= SEQUENCE {
        ptpbaseSystemDomainClockTypeIndex ClockType,
        ptpbaseSystemDomainTotals         Unsigned32
}

ptpbaseSystemDomainClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockProfileType
    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
        "1588 Version 2.0 Section 8.2.2 currentDS data set member
        specifications of [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockCurrentDSDomainIndex,
                        ptpbaseClockCurrentDSClockTypeIndex,
                        ptpbaseClockCurrentDSInstanceIndex
                    }
    ::= { ptpbaseClockCurrentDSTable 1 }

PtpbaseClockCurrentDSEntry ::= SEQUENCE {
        ptpbaseClockCurrentDSDomainIndex      ClockDomainType,
        ptpbaseClockCurrentDSClockTypeIndex   ClockType,
        ptpbaseClockCurrentDSInstanceIndex    ClockInstanceType,
        ptpbaseClockCurrentDSStepsRemoved     Unsigned32,
        ptpbaseClockCurrentDSOffsetFromMaster ClockTimeInterval,
        ptpbaseClockCurrentDSMeanPathDelay    ClockTimeInterval
}

ptpbaseClockCurrentDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockCurrentDSEntry 1 }

ptpbaseClockCurrentDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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       "1588 Version 2.0 Section 8.2.2.2 stepsRemoved"
    ::= { ptpbaseClockCurrentDSEntry 4 }

ptpbaseClockCurrentDSOffsetFromMaster OBJECT-TYPE
    SYNTAX          ClockTimeInterval
    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       "1588 Version 2.0 Section 8.2.2.3 of
                     [IEEE 1588-2008]"
    ::= { ptpbaseClockCurrentDSEntry 5 }

ptpbaseClockCurrentDSMeanPathDelay OBJECT-TYPE
    SYNTAX          ClockTimeInterval
    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 measure by the
        delay request-response mechanism."
    REFERENCE       "1588 Version 2.0 Section 8.2.2.4 mean path delay"
    ::= { 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            ClockDomainType,
    ptpbaseClockParentDSClockTypeIndex         ClockType,
    ptpbaseClockParentDSInstanceIndex          ClockInstanceType,
    ptpbaseClockParentDSParentPortIdentity     OCTET STRING,
    ptpbaseClockParentDSParentStats            TruthValue,
    ptpbaseClockParentDSOffset                 ClockIntervalBase2,
    ptpbaseClockParentDSClockPhChRate          Integer32,
    ptpbaseClockParentDSGMClockIdentity        ClockIdentity,
    ptpbaseClockParentDSGMClockPriority1       Unsigned32,
    ptpbaseClockParentDSGMClockPriority2       Unsigned32,
    ptpbaseClockParentDSGMClockQualityClass    ClockQualityClassType,
    ptpbaseClockParentDSGMClockQualityAccuracy ClockQualityAccuracyType,
    ptpbaseClockParentDSGMClockQualityOffset   Unsigned32
}

ptpbaseClockParentDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockParentDSEntry 1 }

ptpbaseClockParentDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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          ClockIntervalBase2 (-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 clocks 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 clocks 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          ClockIdentity
    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          ClockQualityClassType (0..255)
    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          ClockQualityAccuracyType
    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     ClockDomainType,
        ptpbaseClockDefaultDSClockTypeIndex  ClockType,
        ptpbaseClockDefaultDSInstanceIndex   ClockInstanceType,
        ptpbaseClockDefaultDSTwoStepFlag     TruthValue,
        ptpbaseClockDefaultDSClockIdentity   ClockIdentity,
        ptpbaseClockDefaultDSPriority1       Unsigned32,
        ptpbaseClockDefaultDSPriority2       Unsigned32,
        ptpbaseClockDefaultDSSlaveOnly       TruthValue,
        ptpbaseClockDefaultDSQualityClass    ClockQualityClassType,
        ptpbaseClockDefaultDSQualityAccuracy ClockQualityAccuracyType,
        ptpbaseClockDefaultDSQualityOffset   Integer32
}
```

```
ptpbaseClockDefaultDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockDefaultDSEntry 1 }

ptpbaseClockDefaultDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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          ClockIdentity
    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          ClockQualityClassType (0..255)
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the default dataset Quality Class."
    ::= { ptpbaseClockDefaultDSEntry 9 }

ptpbaseClockDefaultDSQualityAccuracy OBJECT-TYPE
    SYNTAX          ClockQualityAccuracyType
    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     ClockDomainType,
        ptpbaseClockRunningClockTypeIndex  ClockType,
        ptpbaseClockRunningInstanceIndex   ClockInstanceType,
        ptpbaseClockRunningState           ClockStateType,
        ptpbaseClockRunningPacketsSent     Counter64,
        ptpbaseClockRunningPacketsReceived Counter64
}

ptpbaseClockRunningDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockRunningEntry 1 }

ptpbaseClockRunningClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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          ClockStateType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Clock state returned by PTP engine
        which was described earlier.

        Freerun state.  Applies to a slave device that is not locked to
        a master.  This is the initial state a slave starts out with
        when it is not getting any PTP packets from the master, or
        because of some other input error (erroneous packets, etc).

        Holdover state.  In this state the slave device is locked to a
        master but communication with the master has been lost or the
        timestamps in the ptp packets are incorrect.  Since the
        slave was previously locked to the master, it can run in this
        state, with similar accuracy for some time.  If communication
        with the master is not restored for an extended period
        (dependent on the clock implementation), the device should move
        to the FREERUN state.

        Acquiring state. The slave device is receiving packets from a
        master and is trying to acquire a lock.

        Freq_locked state. Slave device is locked to the Master with
        respect to frequency, but not phase aligned

        Phase_aligned state. Locked to the master with respect to
        frequency and phase."
    ::= { ptpbaseClockRunningEntry 4 }

ptpbaseClockRunningPacketsSent OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the total number of all packet Unicast
        and multicast that have been sent out for this clock in this
        domain for this type."
    ::= { ptpbaseClockRunningEntry 5 }

ptpbaseClockRunningPacketsReceived OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the total number of all packet Unicast
        and multicast that have been received for this clock in this
        domain for this type."
    ::= { 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 of [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockTimePropertiesDSDomainIndex,
                        ptpbaseClockTimePropertiesDSClockTypeIndex,
                        ptpbaseClockTimePropertiesDSInstanceIndex
                    }
    ::= { ptpbaseClockTimePropertiesDSTable 1 }

PtpbaseClockTimePropertiesDSEntry ::= SEQUENCE {
  ptpbaseClockTimePropertiesDSDomainIndex           ClockDomainType,
  ptpbaseClockTimePropertiesDSClockTypeIndex        ClockType,
  ptpbaseClockTimePropertiesDSInstanceIndex         ClockInstanceType,
  ptpbaseClockTimePropertiesDSCurrentUTCOffsetValid TruthValue,
  ptpbaseClockTimePropertiesDSCurrentUTCOffset      Integer32,
  ptpbaseClockTimePropertiesDSLeap59                TruthValue,
  ptpbaseClockTimePropertiesDSLeap61                TruthValue,
  ptpbaseClockTimePropertiesDSTimeTraceable         TruthValue,
  ptpbaseClockTimePropertiesDSFreqTraceable         TruthValue,
  ptpbaseClockTimePropertiesDSPTPTimescale          TruthValue,
  ptpbaseClockTimePropertiesDSSource                ClockTimeSourceType
}

ptpbaseClockTimePropertiesDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockTimePropertiesDSEntry 1 }

ptpbaseClockTimePropertiesDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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 current UTC offset is valid."
    REFERENCE       "Section 8.2.4.2 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
        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.
        The initialization value shall be selected as follows:
        a) If the timePropertiesDS.ptpTimescale (see 8.2.4.8) is TRUE,
        the value is the value obtained from a
        primary reference if the value is known at the time of
        initialization, else,
        b) The value shall be the current number of leap seconds (7.2.3)
        when the node is designed."
    REFERENCE       "Section 8.2.4.3 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 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 of [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 7 }

ptpbaseClockTimePropertiesDSTimeTraceable OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Timetraceable value in the clock
        Current Dataset."
    REFERENCE       "Section 8.2.4.6 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 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 of [IEEE 1588-2008]"
    ::= { ptpbaseClockTimePropertiesDSEntry 10 }

ptpbaseClockTimePropertiesDSSource OBJECT-TYPE
    SYNTAX          ClockTimeSourceType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the Timesource value in the clock Current
        Dataset."
    REFERENCE       "Section 8.2.4.9 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 of [IEEE 1588-2008]"
    INDEX           {
                        ptpbaseClockTransDefaultDSDomainIndex,
                        ptpbaseClockTransDefaultDSInstanceIndex
                    }
    ::= { ptpbaseClockTransDefaultDSTable 1 }

PtpbaseClockTransDefaultDSEntry ::= SEQUENCE {
        ptpbaseClockTransDefaultDSDomainIndex   ClockDomainType,
        ptpbaseClockTransDefaultDSInstanceIndex ClockInstanceType,
        ptpbaseClockTransDefaultDSClockIdentity ClockIdentity,
        ptpbaseClockTransDefaultDSNumOfPorts    Counter32,
        ptpbaseClockTransDefaultDSDelay         ClockMechanismType,
        ptpbaseClockTransDefaultDSPrimaryDomain ClockDomainType
}

ptpbaseClockTransDefaultDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockTransDefaultDSEntry 1 }

ptpbaseClockTransDefaultDSInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    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          ClockIdentity
    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 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."
    REFERENCE       "Section 8.3.2.2.2 of [IEEE 1588-2008]"
    ::= { ptpbaseClockTransDefaultDSEntry 4 }

ptpbaseClockTransDefaultDSDelay OBJECT-TYPE
    SYNTAX          ClockMechanismType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object, if the transparent clock is an end-to-end
        transparent clock, has the value shall be E2E; If the
        transparent clock is a peer-to-peer transparent clock, the
        value
        shall be P2P."
    REFERENCE       "Section 8.3.2.3.1 of [IEEE 1588-2008]"
    ::= { ptpbaseClockTransDefaultDSEntry 5 }

ptpbaseClockTransDefaultDSPrimaryDomain OBJECT-TYPE
    SYNTAX          ClockDomainType
    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 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            ClockDomainType,
        ptpbaseClockPortClockTypeIndex         ClockType,
        ptpbaseClockPortClockInstanceIndex     ClockInstanceType,
        ptpbaseClockPortTablePortNumberIndex   ClockPortNumber,
        ptpbaseClockPortName                   DisplayString,
        ptpbaseClockPortRole                   ClockRoleType,
        ptpbaseClockPortSyncTwoStep            TruthValue,
        ptpbaseClockPortCurrentPeerAddressType AutonomousType,
        ptpbaseClockPortCurrentPeerAddress
ClockPortTransportTypeAddress,
        ptpbaseClockPortNumOfAssociatedPorts   Gauge32
}

ptpbaseClockPortDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockPortEntry 1 }

ptpbaseClockPortClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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          ClockPortNumber
    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
        router."
    ::= { ptpbaseClockPortEntry 5 }

ptpbaseClockPortRole OBJECT-TYPE
    SYNTAX          ClockRoleType
    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          ClockPortTransportTypeAddress
    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          ClockDomainType,
        ptpbaseClockPortDSClockTypeIndex       ClockType,
        ptpbaseClockPortDSClockInstanceIndex   ClockInstanceType,
        ptpbaseClockPortDSPortNumberIndex      ClockPortNumber,
        ptpbaseClockPortDSName                 DisplayString,
        ptpbaseClockPortDSPortIdentity         OCTET STRING,
        ptpbaseClockPortDSlogAnnouncementInterval ClockIntervalBase2,
        ptpbaseClockPortDSAnnounceRctTimeout   Integer32,
        ptpbaseClockPortDSlogSyncInterval      ClockIntervalBase2,
        ptpbaseClockPortDSMinDelayReqInterval  Integer32,
        ptpbaseClockPortDSPeerDelayReqInterval Integer32,
        ptpbaseClockPortDSDelayMech            ClockMechanismType,
        ptpbaseClockPortDSPeerMeanPathDelay    ClockTimeInterval,
        ptpbaseClockPortDSGrantDuration        Unsigned32,
        ptpbaseClockPortDSPTPVersion           Unsigned32
}

ptpbaseClockPortDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockPortDSEntry 1 }

ptpbaseClockPortDSClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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          ClockPortNumber
    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          ClockIntervalBase2
    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          ClockIntervalBase2
    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          ClockMechanismType
    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          ClockTimeInterval
    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 dataset for
        a particular domain."
    ::= { ptpbaseMIBClockInfo 9 }

ptpbaseClockPortRunningEntry OBJECT-TYPE
    SYNTAX          PtpbaseClockPortRunningEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the table, containing runing dataset information
        about a single clock port."
    INDEX           {
                        ptpbaseClockPortRunningDomainIndex,
                        ptpbaseClockPortRunningClockTypeIndex,
                        ptpbaseClockPortRunningClockInstanceIndex,
                        ptpbaseClockPortRunningPortNumberIndex
                    }
    ::= { ptpbaseClockPortRunningTable 1 }

PtpbaseClockPortRunningEntry ::= SEQUENCE {
        ptpbaseClockPortRunningDomainIndex        ClockDomainType,
        ptpbaseClockPortRunningClockTypeIndex     ClockType,
        ptpbaseClockPortRunningClockInstanceIndex ClockInstanceType,
        ptpbaseClockPortRunningPortNumberIndex    ClockPortNumber,
        ptpbaseClockPortRunningName               DisplayString,
        ptpbaseClockPortRunningState              ClockPortState,
        ptpbaseClockPortRunningRole               ClockRoleType,
        ptpbaseClockPortRunningInterfaceIndex     InterfaceIndexOrZero,
        ptpbaseClockPortRunningTransport          AutonomousType,
        ptpbaseClockPortRunningEncapsulationType  AutonomousType,
        ptpbaseClockPortRunningTxMode             ClockTxModeType,
        ptpbaseClockPortRunningRxMode             ClockTxModeType,
        ptpbaseClockPortRunningPacketsReceived    Counter64,
        ptpbaseClockPortRunningPacketsSent        Counter64
}

ptpbaseClockPortRunningDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
```

```
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockPortRunningEntry 1 }

ptpbaseClockPortRunningClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    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          ClockInstanceType
    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          ClockPortNumber
    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          ClockPortState
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies the port state returned by PTP engine.

        initializing - In this state a port initializes
                       its data sets, hardware, and
                       communication facilities.
        faulty       - The fault state of the protocol.
        disabled     - The port shall not place any
                       messages on its communication path.
        listening    - The port is waiting for the
                       announceReceiptTimeout to expire or
                       to receive an Announce message from
                       a master.
        preMaster    - The port shall behave in all respects
                       as though it were in the MASTER state
                       except that it shall not place any
                       messages on its communication path
                       except for Pdelay_Req, Pdelay_Resp,
                       Pdelay_Resp_Follow_Up, signaling, or
                       management messages.
        master       - The port is behaving as a master port.
        passive      - The port shall not place any
                       messages on its communication path
                       except for Pdelay_Req, Pdelay_Resp,
                       Pdelay_Resp_Follow_Up, or signaling
                       messages, or management messages
                       that are a required response to
                       another management message
        uncalibrated - The local port is preparing to
                       synchronize to the master port.
        slave        - The port is synchronizing to the
                       selected master port."
    ::= { ptpbaseClockPortRunningEntry 6 }

ptpbaseClockPortRunningRole OBJECT-TYPE
    SYNTAX          ClockRoleType
    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 router 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 (eg. VLAN, Pseudowire
        encapsulation...) for the PTP messages."
    ::= { ptpbaseClockPortRunningEntry 10 }

ptpbaseClockPortRunningTxMode OBJECT-TYPE
    SYNTAX          ClockTxModeType
    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          ClockTxModeType
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifie 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
        (cummulative)."
    ::= { 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
        (cummulative)."
    ::= { 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        ClockDomainType,
        ptpbaseClockPortTransDSInstanceIndex      ClockInstanceType,
        ptpbaseClockPortTransDSPortNumberIndex    ClockPortNumber,
        ptpbaseClockPortTransDSPortIdentity       ClockIdentity,
        ptpbaseClockPortTransDSlogMinPdelayReqInt ClockIntervalBase2,
        ptpbaseClockPortTransDSFaultyFlag         TruthValue,
        ptpbaseClockPortTransDSPeerMeanPathDelay  ClockTimeInterval
}

ptpbaseClockPortTransDSDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the domain number used to create logical
        group of PTP devices."
    ::= { ptpbaseClockPortTransDSEntry 1 }

ptpbaseClockPortTransDSInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    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          ClockPortNumber
    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 [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 3 }

ptpbaseClockPortTransDSPortIdentity OBJECT-TYPE
    SYNTAX          ClockIdentity
    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 of [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 4 }

ptpbaseClockPortTransDSlogMinPdelayReqInt OBJECT-TYPE
    SYNTAX          ClockIntervalBase2
    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 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 of [IEEE 1588-2008]"
    ::= { ptpbaseClockPortTransDSEntry 6 }

ptpbaseClockPortTransDSPeerMeanPathDelay OBJECT-TYPE
    SYNTAX          ClockTimeInterval
    UNITS           "Time Interval"
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object specifies, (if the delayMechanism used is P2P) the
        value is 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 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 which 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           ClockDomainType,
        ptpClockPortCurrentClockTypeIndex        ClockType,
        ptpClockPortCurrentClockInstanceIndex    ClockInstanceType,
        ptpClockPortCurrentPortNumberIndex       ClockPortNumber,
        ptpbaseClockPortAssociatePortIndex       Unsigned32,
        ptpbaseClockPortAssociateAddressType     AutonomousType,
        ptpbaseClockPortAssociateAddress
ClockPortTransportTypeAddress,
        ptpbaseClockPortAssociatePacketsSent     Counter64,
        ptpbaseClockPortAssociatePacketsReceived Counter64,
        ptpbaseClockPortAssociateInErrors        Counter64,
        ptpbaseClockPortAssociateOutErrors       Counter64
}

ptpClockPortCurrentDomainIndex OBJECT-TYPE
    SYNTAX          ClockDomainType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the given port’s domain number."
    ::= { ptpbaseClockPortAssociateEntry 1 }

ptpClockPortCurrentClockTypeIndex OBJECT-TYPE
    SYNTAX          ClockType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the given port’s clock type."
    ::= { ptpbaseClockPortAssociateEntry 2 }

ptpClockPortCurrentClockInstanceIndex OBJECT-TYPE
    SYNTAX          ClockInstanceType
    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          ClockPortNumber
    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."
    ::= { ptpbaseClockPortAssociateEntry 6 }

ptpbaseClockPortAssociateAddress OBJECT-TYPE
    SYNTAX          ClockPortTransportTypeAddress
    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."
    ::= { 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."
    ::= { 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."
    ::= { 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."
    ::= { 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
        Dataset
        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

This MIB contains readable objects whose values provide information related to PTP objects. It does not contain writable objects.

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.

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.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework (see [RFC 3410], section 8). Specifically, the use of the User-based Security Model [RFC 3414] and the View-based Access Control Model [RFC 3415] is recommended.

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 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 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 5905] David L. Mills, " Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, University of Delaware, June 2010.

[IEEE 802.3-2008] "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and Metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE Std. 802.3 - 2008, 26 December 2008

[G.8265.1] "Precision time protocol telecom profile for frequency synchronization", ITU-T Recommendation G.8265.1, October 2010.

8. Acknowledgements

Thanks to John Linton and Danny Lee for valuable comments, and to Bert Wijnen, Kevin Gross, Alan Luchuk and Chris Elliot for their reviews of this MIB.

9. Author’s Addresses

Vinay Shankarkumar
Cisco Systems,
7025-4 Kit Creek Road,
Research Triangle Park,
NC 27560,
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 N. First Str.,
San Jose,
CA 95134,
USA.

Email: greg.dowd@microsemi.com

Internet Working Group   Y. Jiang
                                                                  X. Liu
Internet Draft   J. Xu
                                                                  Huawei
Intended status: Standards Track   R. Cummings
                                                    National Instruments
Expires: April 2015   October 16, 2015

YANG Data Model for IEEE 1588v2
draft-jlx-tictoc-1588v2-yang-02.txt

Status of this Memo

Copyright Notice

Abstract

This document defines a YANG data model for the configuration of IEEE 1588v2 devices and clocks, and also retrieval of the configuration information, data set and running states of IEEE 1588v2 clocks.

Table of Contents

1. Introduction

As a synchronization protocol, IEEE 1588v2 [IEEE1588] is widely supported in the carrier networks. It can provide high precision time synchronization as high as nano-seconds. The protocol depends on a Precision Time Protocol (PTP) engine to automatically decide its state, and a PTP transportation layer to carry the PTP timing and various quality messages. The configuration parameters and state data sets of IEEE 1588v2 are numerous.

Some work on IEEE 1588v2 MIB [PTP-MIB] is in progress in the IETF TICTOC WG. But the work is only scoped with retrieval of the state data of IEEE 1588v2 by Simple Network Management Protocol (SNMP) and configuration is not considered, thus its use is limited.

Some service providers require the management of the IEEE 1588v2 synchronization network can be more flexible and more Internet-based (typically overlaid on their transport networks). Software Defined Network (SDN) is another driving factor which demands a greater control over synchronization networks.

YANG [RFC6020] is a data modeling language used to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF) [RFC6241]. A small set of built-in data types are defined in [RFC6020], and a collection of common data types are further defined in [RFC6991]. Advantages of YANG include Internet based configuration capability, validation, roll-back and etc., all these characteristics make it attractive to become a modeling language for IEEE 1588v2.

This document defines a YANG [RFC6020] data model for the configuration of IEEE 1588v2 devices and clocks, and also retrieval of the state data of IEEE 1588v2 clocks.

In order to fulfill the need of a lightweight implementation, the core module is designed to be generic and minimal, but be extensible with capability negotiation. That is, if a node is verified with a capability of more functions, then more modules can be loaded on demand, otherwise, only a basic module is loaded on the node.

This document defines PTP system information, PTP data sets and running states following the structure and definitions in IEEE 1588v2, and compatible with [PTP-MIB]. The router specific 1588v2 information is out of scope of this document.

2. Conventions used in this document

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].

3. Terminology

Terminologies used in this document are extracted from [IEEE1588] and [PTP-MIB].

ARB	Arbitrary Timescale
BC	Boundary Clock
DS	Data Set
E2E	End-to-End
EUI	Extended Unique Identifier.
GPS	Global Positioning System

IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OC	Ordinary Clock
P2P	Peer-to-Peer
PTP	Precision Time Protocol
TAI	International Atomic Time
TC	Transparent Clock
UDP	User Datagram Protocol
UTC	Coordinated Universal Time

4. IEEE 1588V2 YANG Model hierarchy

This section describes the hierarchy of IEEE 1588v2 YANG module. Query and retrieval of device wide or port specific configuration information and clock data set is described for this version.

Query and retrieval of clock information include:

- Clock data set attributes in a clock node, including: current-DS, clock-parent-DS, default-DS, time-properties-DS, and transparentClock-default-DS.

- Port specific data set attributes, including: port-DS and transparentClock-port-DS.

```
   module: ietf-yang-ptp-dataset
      +--rw  ptp-datasets* [domain-number]
       +--rw domain-number   uint8
       +--rw default-DS
       |  +--rw two-step-flag?    boolean
       |  +--rw clock-identity?   binary
       |  +--rw number-ports?     uint16
       |  +--rw clock-quality
       |  |  +--rw clock-class?                  uint8
       |  |  +--rw clock-accuracy?               uint8
       |  |  +--rw offset-scaled-log-variance?   uint16
       |  +--rw priority1?        uint8
       |  +--rw priority2?        uint8
       |  +--rw slave-only?       boolean
       +--rw current-DS
       |  +--rw steps-removed?        uint16
       |  +--rw offset-from-master?   binary
       |  +--rw mean-path-delay?      binary
       +--rw parent-DS
       |  +--rw parent-port-identity
       |  |  +--rw clock-identity?   binary
       |  |  +--rw port-number?      uint32
       |  +--rw parent-stats?                              boolean
         |  +--rw observed-parent-offset-scaled-log-variance?   uint16
         |  +--rw observed-parent-clock-phase-change-rate?      int32
         |  +--rw grandmaster-identity?                         binary
         |  +--rw grandmaster-clock-quality
         |  |  +--rw grandmaster-clock-class?                  uint8
         |  |  +--rw grandmaster-clock-accuracy?               uint8
         |  |  +--rw grandmaster-offset-scaled-log-variance?   uint16
         |  +--rw grandmaster-priority1?                        uint8
         |  +--rw grandmaster-priority2?                        uint8
         +--rw time-properties-DS
         |  +--rw current-UTC-offset-valid?   boolean
         |  +--rw current-UTC-offset?         uint16
         |  +--rw leap59?                     boolean
         |  +--rw leap61?                     boolean
         |  +--rw time-traceable?             boolean
         |  +--rw frequency-traceable?        boolean
         |  +--rw PTP-timescale?              boolean
         |  +--rw time-source?                uint8
         +--rw port-DS-list* [port-number]
         |  +--rw port-number                    uint32
         |  +--rw port-identity
         |  |  +--rw clock-identity?   binary
         |  |  +--rw port-number?      uint32
         |  +--rw port-state?                    uint8
         |  +--rw log-min-delay-req-interval?    int8
         |  +--rw peer-mean-path-delay?          int64
         |  +--rw log-announce-interval?         int8
         |  +--rw announce-receipt-timeout?      uint8
         |  +--rw log-sync-interval?             int8
         |  +--rw delay-mechanism?               enumeration
         |  +--rw log-min-Pdelay-req-interval?   int8
         |  +--rw version-number?                uint8
         +--rw transparent-clock-default-DS
         |  +--rw clock-identity?    binary
         |  +--rw number-ports?      uint16
         |  +--rw delay-mechanism?   enumeration
         |  +--rw primary-domain?    uint32
         +--rw transparent-clock-port-DS-list* [port-number]
            +--rw port-number                    uint32
            +--rw port-identity
            |  +--rw clock-identity?   binary
            |  +--rw port-number?      uint32
            +--rw log-min-Pdelay-req-interval?   int8
            +--rw faulty-flag?                   boolean
            +--rw peer-mean-path-delay?          int64
```

5. IEEE 1588v2 YANG Module

```
   module ietf-yang-ptp-dataset {
     namespace "urn:ietf:params:xml:ns:yang:1588v2";
     prefix "ptp-dataset";
     organization "IETF TICTOC WG";
     contact "jiangyuanlong@huawei.com";
     description
          "This YANG module defines a data model for the configuration
          of IEEE 1588v2 devices and clocks, and also retrieval of the
          state data of IEEE 1588v2 clocks.";
     revision "2015-10-15"{
       description "Initial revision.";
        reference "draft-jxl-tictoc-1588v2-yang";
      }

     grouping default-DS-entry {
       description
             "This group bundles together all information about the
             PTP clock Default Datasets for a single device.";

       leaf two-step-flag {
         description
               "This object specifies whether the Two Step process is
               used.";
         type boolean;
       }
       leaf clock-identity {
         description
               "This object specifies the clockIdentity of the local
               clock";
         config true;
         type binary {
           length "8";
         }
       }

       leaf number-ports {
         description
               "This object specifies the number of PTP ports on the
               device.";
         type uint16;
       }

       container clock-quality {
         description
```

```
               "This object specifies the default clockQuality of the
               device, which contains clockClass, clockAccuracy and
               offsetScaledLogVariance.";

         leaf clock-class {
           description
                 "This object specifies the Quality Class in the
                 defaultDS.";
           type uint8;
         }
         leaf clock-accuracy {
           description
                 "This object specifies the Quality Accuracy in the
                 defaultDS.";
           type uint8;
         }
         leaf offset-scaled-log-variance {
           description
                 "This object specifies the Quality offset in the
                 defaultDS.";
           type uint16;
         }
       }

       leaf priority1 {
         description
               "This object specifies the clock Priority1 in the
               defaultDS ";
         type uint8;
       }
       leaf priority2{
         description
               "This object specifies the clock Priority2 in the
               defaultDS ";
         type uint8;
       }


       leaf slave-only {
         description
               "This object indicates whether the SlaveOnly flag is
               set";
         type boolean;
       }
     }

     grouping current-DS-entry {
       description
             "This group bundles together all information about the
             PTP clock Current Datasets for a single device.";

       leaf steps-removed {
            description
               "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.";
          type uint16;
          default 0;
       }
       leaf offset-from-master {
          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.";
          type binary {
            length "1..255";
          }
       }
       leaf mean-path-delay {
          description
               "The mean path delay between a pair of ports as measured
               by the delay request-response mechanism.";
          type binary {
            length "1..255";
          }
       }
     }

     grouping parent-DS-entry {
       description
             "This group bundles together all information about the
             PTP clock Parent Datasets for a single device.";

       leaf parent-port-identity {
          description
               "This object specifies the value of portIdentity of the
               port on the master that issues the Sync messages used in
               synchronizing this clock.";

         leaf clock-identity {
           description
                 "This object identifies the clockIdentity of the
                 master clock.";
           type binary {
             length "8";
           }
         }

         leaf port-number {
           description
                 "This object identifies the PortNumber for the port
                 on the specific master.";
           type uint32{
             range "0..65535";
           }
         }
              }
       leaf parent-stats {
          description
               "This object indicates whether the values of
               parentDS.observedParentOffsetScaledLogVariance and
               parentDS.observedParentClockPhaseChangeRate have been
               measured and are valid.";
          type boolean;
          default false;
       }
       leaf observed-parent-offset-scaled-log-variance {
          description
               "This object specifies an estimate of the parent clock’s
               phase change rate as measured by the slave clock.";

         type uint16;
          default 0xFFFF;
       }
       leaf observed-parent-clock-phase-change-rate {
          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.";
          type int32;
       }
       leaf grandmaster-identity {
          description
               "This object specifies the clockIdentity of the
               Grandmaster clock.";
          type binary{
            length "8";
          }
       }
       container grandmaster-clock-quality {
         description
               "This object specifies the clockQuality of the
               grandmaster clock.";

         leaf grandmaster-clock-class {
           description
                 "This object specifies the Quality Class of
                 grandmaster clock.";
           type uint8;
         }
         leaf grandmaster-clock-accuracy {
           description
                 "This object specifies the Quality Accuracy of the
                 grandmaster clock.";
           type uint8;
         }
         leaf grandmaster-offset-scaled-log-variance {
           description
                 "This object specifies the Quality offset of the
                 grandmaster clock.";
           type uint16;
         }
       }
       leaf grandmaster-priority1 {
          description
               "This object specifies the priority1 attribute of the
               grandmaster clock.";
          type uint8;
       }
       leaf  grandmaster-priority2 {
          description
               "This object specifies the priority2 attribute of the
               grandmaster clock.";
          type uint8;
       }

     }



     grouping time-properties-DS-entry {
       description
               "This group bundles together all information about the
               PTP clock time properties datasets for a single device.";

       leaf current-UTC-offset-valid {
          description
               "This object indicates whether current UTC offset is
               valid.";
          type boolean;
       }
       leaf current-UTC-offset {
          description
               "This object specifies the offset between TAI and UTC
               when the epoch of the PTP system is the PTP epoch,
               otherwise the value has no meaning. The value shall be
               in units of seconds.";
          type uint16;
       }
       leaf leap59 {
          description
               "This object indicates whether the last minute of the
               current UTC day contains 59 seconds.";
          type boolean;
       }
       leaf leap61 {
          description
               "This object indicates whether the last minute of the
               current UTC day contains 61 seconds.";
          type boolean;
       }
       leaf time-traceable {
          description
               "This object indicates whether the timescale and the
               currentUtcOffset are traceable to a primary reference.";
          type boolean;
       }
       leaf frequency-traceable {
          description
               "This object indicates whether the frequency determining
               the timescale is traceable to a primary reference.";
          type boolean;
       }
       leaf PTP-timescale {
          description
               "This object indicates whether the clock timescale of
               the grandmaster clock is PTP.";
          type boolean;
       }
       leaf time-source {
          description
               "This object specifies the source of time used by the
               grandmaster clock.";
```

```
         type uint8;
       }
     }

      grouping port-DS-entry {
       description
               "This group bundles together all information about the
               clock ports dataset for a single clock port.";

       container port-identity {
          description
               "This object specifies the PTP clock port Identity,
               composed of clock-identity and portNumber.";
          leaf clock-identity {
            description
              "This object identify a specific PTP node.";
            config true;
            type binary {
              length "8";
            }
          }

          leaf port-number {
            description
                 "This object specifies the PTP Portnumber for this
                 port.";
            type uint32 {
              range "0..65535";
            }
          }
       }

       leaf port-state {
         description
               "This object specifies the current state of the protocol
               engine associated with the port.";
         type uint8;
         default 1;
       }

       leaf log-min-delay-req-interval {
         description
               "This object specifies the Delay_Req message
               transmission interval.";
         type int8;
       }
       leaf peer-mean-path-delay {
         description
               "This object specifies an estimate of the current one-
               way propagation delay on the link when the
               delayMechanism is P2P, otherwise it shall be zero.";
         type int64;
         default 0;
       }

       leaf log-announce-interval {
          description
               "This object specifies the Announce message transmission
               interval associated with this clock port.";
          type int8;
       }

       leaf announce-receipt-timeout {
          description
               "This object specifies the number of announceInterval
               that have to pass without receipt of an Announce message
               before the occurrence of the event
               ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES.";
          type uint8;
       }

       leaf log-sync-interval {
          description
               "This object specifies the mean time interval between
               successive Sync messages.";
          type int8;
       }

       leaf delay-mechanism {
          description
               "This object specifies the delay measuring mechanism
               used by the port. If the clock is an end-to-end clock,
               the value is e2e, else if the clock is a peer-to-peer
               clock, the value shall be p2p.";
          type enumeration {
            enum E2E {
              value 01;
              description
                    "The port is configured to use the delay request-
                    response mechanism.";
            }
            enum P2P {
              value 02;
              description
                    "The port is configured to use the peer delay
                    mechanism.";
            }
            enum DISABLED {
              value 254;
              description
                    "The port does not implement the delay mechanism.";
            }
          }
       }

       leaf log-min-Pdelay-req-interval {
         description
               "This object specifies the minimum permitted mean time
               interval between successive Pdelay_Req messages.";
         type int8;
       }

       leaf version-number {
          description
               "This object specifies the version of this standard
               implemented on the port.";
          type uint8;
       }
     }

     grouping transparent-clock-default-DS-entry {
       description
               "This group bundles together the default data sets of a
               transparent clock.";
       leaf clock-identity {
         description
               "This object specifies the value of the clockIdentity
               attribute of the local clock.";
         type binary {
           length "8";
         }
       }
       leaf number-ports {
         description
               "This object specifies the number of PTP ports of the
               device.";
         type uint16;
       }
       leaf delay-mechanism {
         description
               "This object specifies the delayMechanism of the
               transparent clock.";
         type enumeration {
           enum E2E {
             value 1;
             description
                   "The port is configured to use the delay request-
                   response mechanism.";
           }
           enum P2P {
             value 2;
             description
                   "The port is configured to use the peer delay
                   mechanism.";
           }
           enum DISABLED {
             value 254;
             description
                   "The port does not implement the delay mechanism.";
           }
         }
       }
       leaf primary-domain {
         description
               "This object specifies the value of the primary
               syntonization domain.";
         type uint32 {
           range "0..255";
         }
         default 0;
       }
     }

     grouping transparent-clock-port-DS-entry {
       description
               "This group bundles together the port data sets of a
               transparent clock.";

       container port-identity {
         description
               "This object specifies the portIdentity of the local
               port.";
         leaf clock-identity {
           config true;
           type binary {
             length "8";
           }
         }

         leaf port-number {
           type uint32 {
             range "0..65535";
           }
         }
       }
       leaf log-min-Pdelay-req-interval {
          description
               "This object specifies the minimum permitted mean time
               interval between successive Pdelay_Req messages.";
         type int8;
       }
       leaf faulty-flag {
          description
               "This object indicates whether the port is faulty.";
          type boolean;
          default false;
       }
       leaf peer-mean-path-delay {
          description
               "This object specifies an estimate of the current one-
               way propagation delay on the link when the
               delayMechanism is P2P, otherwise it shall be zero.";

         type int64;
          default 0;
       }
     }

     list ptp-datasets {

       key "domain-number";
       min-elements "1";

       description
             "List of one or more PTP datasets in the device, one for
             each domain-number (see IEEE 1588-2008 subclause 6.3)";

       leaf domain-number {
         type uint8;
       }

        container default-DS {
          uses default-DS-entry;
        }

        container current-DS {
          uses current-DS-entry;
        }

        container parent-DS {
          uses parent-DS-entry;
        }

        container time-properties-DS {
          uses time-properties-DS-entry;
        }


         list port-DS-list {
          key "port-number";
          min-elements "1";
          leaf port-number {
            type uint32 {
              range "0..65535";
            }
          }
          uses port-DS-entry;
        }

        container transparent-clock-default-DS {
          uses transparent-clock-default-DS-entry;
        }

        list transparent-clock-port-DS-list {
          key "port-number";
          min-elements "1";
          leaf port-number {
            type uint32 {
              range "0..65535";
            }
          }
          uses transparent-clock-port-DS-entry;
        }
     }
   }
```

6. Security Considerations

YANG modules are designed to be accessed via the NETCONF protocol [RFC6241], thus security considerations in [RFC6241] apply here. Security measures such as using the NETCONF over SSH [RFC6242] and restricting its use with access control [RFC6536] can further improve its security, avoid injection attacks and misuse of the protocol.

Some data nodes defined in this YANG module are writable, and any changes to them may adversely impact a synchronization network.

7. IANA Considerations

This document registers a URI in the IETF XML registry, and the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:1588v2

This document registers a YANG module in the YANG Module Names:

name: 1588v2 namespace: urn:ietf:params:xml:ns:yang:1588v2

8. References

8.1. Normative References

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) ", RFC 6020, October 2010

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013

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

8.2. Informative References

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012

[PTP-MIB] Shankarkumar, V., Montini, L., Frost, T., and Dowd, G., "Precision Time Protocol Version 2 (PTPv2) Management Information Base", draft-ietf-tictoc-ptp-mib-07, Work in progress

9. Acknowledgments

TBD

Authors' Addresses

Yuanlong Jiang
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China
Email: jiangyuanlong@huawei.com

Xian Liu
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China

Jinchun Xu
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China

Rodney Cummings
National Instruments
Email: Rodney.Cummings@ni.com

Internet Engineering Task Force
Internet-Draft
Intended status: Best Current Practice
Expires: March 21, 2016

D. Reilly
Spectracom Corporation

September 18, 2015

Network Time Protocol Best Current Practices
draft-reilly-ntp-bcp-00

Abstract

NTP Version 4 (NTPv4) has been widely used since its publication as RFC 5905 [RFC5905]. This documentation is a collection of Best Practices from across the NTP community.

Status of This Memo

This Internet-Draft will expire on March 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

NTP Version 4 (NTPv4) has been widely used since its publication as RFC 5905 [RFC5905]. This documentation is a collection of Best Practices from across the NTP community.

1.1. Requirements Language

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

2. Keeping NTP up to date

No software (not even NTP) is perfect. Bugs can be present in any software. As software is widely deployed, users find more bugs. And even if software is thoroughly tested and "all" the bugs are wrung out, users continuously find new ways to use software that their authors did not conceive of, which can uncover more bugs. Thousands of individual bugs have been found and fixed in the NTPv4 reference implementation since the first release in 1997.

In addition, there are always new ideas about security on the Internet, and an application which is secure today could be insecure tomorrow once an unknown bug (or a known behavior) is exploited in the right way. Many security mechanisms rely on time, either directly or indirectly, as part of their operation. If an attacker can spoof the time, they may be able to bypass or neutralize other security elements.

In general, the best way to protect yourself and your networks against these bugs and security threats is to make sure that you keep your NTP implementation up to date. The NTP protocol has many different implementations on many different platforms. It is advised that NTP users actively monitor wherever they get their software to find out when updates are available, and deploy them as soon as practical.

The original implementation of NTP Version 4 is still actively maintained and being developed by Network Time Foundation with help from volunteers. The Network Time Foundation currently maintains the reference implementation for NTP at http://www.ntp.org/downloads.html and also at https://github.com/ntp-project/ntp .

3. General Network Security Best Prectices

NTP deployments are only as secure as the network they are running on.

3.1. BCP 38

Many network attacks rely on modifying the IP source address of a packet to point to a different IP address than the computer which originated it. This behavior has been known for quite some time, and BCP 38 [RFC2827] was approved to address this in 2000. This document calls for filtering incoming traffic to make sure that the source IP address is consistent with the networks that are connected to that interface. It’s recommended that all large networks (and ISP’s of any size) implement this. More information is availavle at http://www.bcp38.info .

4. NTP Configuration Best Practices

NTP can be made more secure by making a few simple changes to the ntp.conf file.

4.1. Mode 7

NTP Mode 7 packets can be used as a vehicle for a Denial of Service Attack. Users can prevent their NTP servers from participating by adding the following to their ntp.conf file:

```
restrict default kod nomodify notrap nopeer noquery

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

4.2. Autokey

Editor’s Note: Someone who is smarter than I am will have to write this one.

4.3. Using Pool Servers

It only takes a small amount of bandwidth to synchronize one NTP client, but NTP servers that can service tens of thousands of clients can take considerable resources to run. Users who want to synchronize their computers should only synchronize to servers that they have permission to use.

The NTP pool project is a collection of volunteers who have donated their compting and bandwidth resources to provide time on the Internet for free. The time is generally of good quality, but comes with no guarantee whatsoever. If you are interested in using the pool, please review their instrutions at http://www.pool.ntp.org/en/use.html .

If you want to synchronize multiple computers using the pool, consider running your own NTP server, synchronizing that to the pool, and synchronizing your clients to your in-house NTP server. This reduces the load on the pool.

4.4. Starting, Cold-Starting, and Re-Starting NTP

Only use -g on cold-start. Other things TBD.

Editor’s Note: I think I’d like to expand this a bit to cover how to deal with NTP stopping, when to restart it, and under what circumstances to not restart it!

5. NTP in Embedded Devices

Readers of this BCP already understand how important accurate time is for network computing. And as computing becomes more ubiquitous, there will be many small "Internet of Things" devices that require accurate time. These embedded devices may not have a traditional user interface, but if they connect to the Internet they will be subject to the same security threats as traditional deployments.

5.1. Updating Embedded Devices

Vendors of embedded devices have a special responsibility to pay attention to the current state of NTP bugs and security issues, because their customers usually don’t have the ability to update their NTP implementation on their own. Those devices may have a single firmware upgrade, provided by the manufacturer, that updates all capabilities at once. This means that the vendor essentially assumes the responsibility of making sure their devices have the latest NTP updates applied.

This should also include the ability to update the NTP server address.

(Note: do we find specific historical instances of devices behaving badly and cite them here?)

5.2. KISS Packets

The "Kiss-o’-Death" packet is a rate limiting mechanism where a server can tell a misbehaving client to "back off" its query rate. It is important for all NTP devices to respect these packets and back off when asked to do so by a server. It is even more important for an embedded device, which may not have exposed a control interface for NTP.

5.3. Server configuration

Vendors of embedded devices that need time synchronization should also carefully consider where they get their time from. There are several public-facing NTP servers available, but they may not be prepared to service requests from thousands of new devices on the Internet.

Vendors are encouraged to invest resources into providing their own time servers for their devices.

5.3.1. Get a vendor subdomain for pool.ntp.org

The NTP Pool Project offers a program where vendors can obtain their own subdomain that is part of the NTP Pool. This offers vendors the ability to safely make use of the time distributed by the Pool for their devices. Vendors are encouraged to support the pool if they participate. For more information, visit http://www.pool.ntp.org/en/vendors.html .

6. NTP Deployment Examples

A few examples of interesting NTP Deployments

6.1. Client-Only configuration

TBD

6.2. Server-Only Configuration

TBD

6.3. Anycast

Anycast is described in BCP 126 [RFC4786]. (Also see RFC 7094 [RFC7094]). With anycast, a single IP address is assigned to multiple interfaces, and routers direct packets to the closest active interface.

Anycast is often used for Internet services at known IP addresses, such as DNS. Anycast can also be used in large organizations to simplify configuration of a large number of NTP clients. Each client can be configured to the same IP address, and a pool of anycast servers can be deployed to service those requests. New servers can be added to or taken from the pool, and other than a temporary loss of service while a server is taken down, these additions can be transparent to the clients.

While clients are connected to an NTP server via anycast, the client does not know which particular server they are connected to. And as anycast servers enter and leave the network, the server a particular client is connected to may change, which can cause temporary problems on the client. It is recommended that anycast is deployed in environments where precision synchronization is not required.

A client also may not have any way to diagnose if an anycast server is not functioning properly. It is recommended that any anycast NTP implementation include multiple interfaces with at least one Unicast address. These Unicast addresses should be monitored (perhaps in a peering arrangement) so that if one server’s reference goes bad, it can use the other servers to validate the correct time.

7. Acknowledgements

The author wishes to acknowledge the contributions of Harlan Stenn, Sue Graves, Samuel Weiler, and Karen O’Donoghue.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBD

10. Normative References

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

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <http://www.rfc-editor.org/info/rfc4786>.

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

[RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <http://www.rfc-editor.org/info/rfc7094>.

Author’s Address

Denis Reilly
Spectracom Corporation
1565 Jefferson Road, Suite 460
Rochester, NY 14623
US

Email: denis.reilly@spectracom.orolia.com

Network Working Group   N. Wu
Internet-Draft   A. Kumar S N
Intended status: Standards Track   Huawei
Expires: December 29, 2015   June 27, 2015

A YANG Data Model for NTP
draft-wu-ntp-ntp-cfg-01

Abstract

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

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for Network Time Protocol [RFC5905] implementations.

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

1.1. Terminology

The following terms are defined in [RFC6020]:

- configuration data

- data model

- module

- state data

The terminology for describing YANG data models is found in [RFC6020].

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. NTP data model

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

```
module: ietf-ntp
   +--rw ntp-cfg!
   |  +--rw ntp-enabled?       boolean
   |  +--rw refclock-master
   |  |  +--rw master?           boolean
   |  |  +--rw master-stratum?   ntp-stratum
   |  +--rw authentication!
   |  |  +--rw auth-enabled?          boolean
   |  |  +--rw trusted-key?           uint32
   |  |  +--rw authentication-keys* [key-id]
   |  |     +--rw key-id       uint32
   |  |     +--rw algorithm?   enumeration
   |  |     +--rw password?    union
   |  +--rw access-rules
   |  |  +--rw access-rule* [access-mode]
   |  |     +--rw access-mode    enumeration
   |  |     +--rw acl-number
   |  |        +--rw (acl-type)?
   |  |           +--:(ipv4)
   |  |           |  +--rw acl-number-ipv4?   uint16
   |  |           +--:(ipv6)
   |  |              +--rw acl-number-ipv6?   uint16
   |  +--rw associations
```

```
      |  |  +--rw peers
      |  |  |  +--rw peer* [address vrf]
      |  |  |     +--rw version?   ntp-version
      |  |  |     +--rw address    inet:ip-address
      |  |  |     +--rw key-id?    leafref
      |  |  |     +--rw minpoll?   ntp-minpoll
      |  |  |     +--rw maxpoll?   ntp-maxpoll
      |  |  |     +--rw prefer?    boolean
      |  |  |     +--rw burst?     boolean
      |  |  |     +--rw iburst?    boolean
      |  |  |     +--rw vrf        string
      |  |  |     +--rw source?    leafref
      |  |  +--rw servers
      |  |     +--rw server* [address vrf]
      |  |        +--rw version?   ntp-version
      |  |        +--rw address    inet:ip-address
      |  |        +--rw key-id?    leafref
      |  |        +--rw minpoll?   ntp-minpoll
      |  |        +--rw maxpoll?   ntp-maxpoll
      |  |        +--rw prefer?    boolean
      |  |        +--rw burst?     boolean
      |  |        +--rw iburst?    boolean
      |  |        +--rw vrf        string
      |  |        +--rw source?    leafref
      |  +--rw ntp-interfaces
      |     +--rw ntp-interface* [ntp-ifname]
      |        +--rw ntp-ifname          leafref
      |        +--rw multicast-client
      |        |  +--rw multicast-client-address?   union
      |        +--rw multicast-server
      |        |  +--rw multicast-server-address?   inet:ip-address
      |        |  +--rw multicast-server-ttl?       uint8
      |        |  +--rw multicast-server-version?   ntp-version
      |        |  +--rw multicast-server-keyid?     leafref
      |        +--rw broadcast-client
      |        |  +--rw broadcast-client-enabled?   boolean
      |        +--rw broadcast-server
      |           +--rw broadcast-server-version?   ntp-version
      |           +--rw broadcast-server-keyid?     leafref
      +--ro ntp-state
         +--ro system-status
         |  +--ro clock-state?       enumeration
         |  +--ro clock-stratum?     ntp-stratum
         |  +--ro clock-refid?       union
         |  +--ro nominal-freq?      decimal64
         |  +--ro actual-freq?       decimal64
         |  +--ro clock-precision?   uint8
         |  +--ro clock-offset?      decimal64
         |  +--ro root-delay?        decimal64
         |  +--ro root-dispersion?   decimal64
         |  +--ro peer-dispersion?   decimal64
         |  +--ro reference-time?    string
         |  +--ro sync-state?        enumeration
         +--ro associations-status
         |  +--ro association-status* [association-source]
         |     +--ro association-source        union
         |     +--ro association-stratum?      ntp-stratum
         |     +--ro association-refid?        union
         |     +--ro association-reach?        uint8
         |     +--ro association-poll?         uint8
         |     +--ro association-now?          uint32
         |     +--ro association-offset?       decimal64
         |     +--ro association-delay?        decimal64
         |     +--ro association-dispersion?   decimal64
         |     +--ro association-sent?         uint32
         |     +--ro association-sent-fail?    uint32
         |     +--ro association-received?     uint32
         |     +--ro association-dropped?      uint32
         +--ro ntp-statistics
            +--ro packet-sent?        uint32
            +--ro packet-sent-fail?   uint32
            +--ro packet-received?    uint32
            +--ro packet-dropped?     uint32
```

This data model defines two primary containers, one for NTP configuration and the other is for NTP running state. The NTP configuration container includes data nodes for access rules, authentication, associations and interfaces. In the NTP running state container, there are data nodes for system status and associations.

3. Relationship to NTPv4-MIB

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

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

YANG data nodes in /ntp-cfg/	NTPv4-MIB objects
ntp-enabled	ntpEntStatusCurrentMode

YANG data nodes in /ntp-cfg/associations/peers/peer /ntp-cfg/associations/servers/server	NTPv4-MIB objects
address	ntpAssocAddressType ntpAssocAddress

YANG NTP Configuration Data Nodes and Related NTPv4-MIB Objects

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

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

YANG NTP State Data Nodes and Related NTPv4-MIB Objects

4. NTP YANG Module

```
   //<CODE BEGINS> file "ietf-ntp@2015-06-27.yang"

   module ietf-ntp {

     namespace "urn:ietf:params:xml:ns:yang:ietf-ntp";

     prefix "ntp";

     import ietf-inet-types {
       prefix "inet";
     }

     import ietf-interfaces {
       prefix "if";
     }

     organization
       "IETF NTP (Network Time Protocol) Working Group";

     contact
       "WG Web:  <http://tools.ietf.org/wg/ntp/>
        WG List:  <mailto: ntpwg@lists.ntp.org
        WG Chair: Karen O'Donoghue
                  <mailto: odonoghue@isoc.org>
        Editor:   Eric Wu
                  <mailto:eric.wu@huawei.com>";
        Editor:   Anil Kumar S N
                  <mailto:anil.sn@huawei.com>";

     description
       "This YANG module defines essential components for the
        management of a routing subsystem.

        Copyright (c) 2014 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

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

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

     revision 2015-06-27 {
       description
```

```
         "Updated revision.";
       reference
         "RFC XXXX: A YANG Data Model for NTP Management";
     }



     /* Typedef Definitions */
     typedef ntp-stratum {
      type uint8;
      description
        "The level of each server in the hierarchy is defined by a
        stratum number.  Primary servers are assigned stratum one;
        secondary servers at each lower level are assigned stratum
        numbers one greater than the preceding level";
     }

     typedef ntp-version {
       type uint8 {
         range "1..4";
       }
       default "3";
       description
         "The current NTP version supported by corresponding
         association.";
     }

     typedef ntp-minpoll {
       type uint8 {
         range "4..17";
       }
       default "6";
       description
         "The minimul poll interval for this NTP association.";
     }


     typedef ntp-maxpoll {
       type uint8 {
         range "4..17";
       }
       default "10";
       description
         "The maximul poll interval for this NTP association.";
     }


     typedef multicast-client-v4address {
       type inet:ipv4-address;
       default "224.0.1.1";
       description
         "The IPv4 address for NTP multicast client.";
     }


     typedef multicast-client-v6address {
       type inet:ipv6-address;
       default "FF0E::0101";
       description
         "The IPv6 address for NTP multicast client.";
     }

     /* Groupings */
     grouping authentication-key {
       description
         "To define an authentication key for a Network Time Protocol
         (NTP) time source.";
       leaf key-id {
         type uint32 {
           range "1..max";
         }
         description
           "Authentication key identifier.";
       }
       leaf algorithm {
         type enumeration {
           enum md5 {
             description
               "Message Digest 5 (MD5) algorithm.";
           }
           enum hmac-sha256 {
             description
               "Secure Hash Algorithm 256 algorithm.";
           }
         }
         description
           "Authentication algorithm.";
       }
       leaf password {
         type union {
           type string {
               length "1..255";
           }
           type string {
               length "20..392";
           }
         }
         description "Clear or encrypted mode for password text.";
       }
     }

     grouping association-param {
       description
         "To define parameters for a Network Time Protocol (NTP)
         associations.";
       leaf version {
         type ntp-version;
         description
           "NTP version.";
       }
       leaf address {
         type inet:ip-address;
         description
           "The IP address of this association.";
       }
       leaf key-id {
         type leafref {
           path "/ntp:ntp-cfg/ntp:authentication/"
                 + "ntp:authentication-keys/ntp:key-id";
         }
         description
           "Authentication key id referenced in this association.";
       }
       leaf minpoll {
         type ntp-minpoll;
         description
           "The minimul poll interval used in this association.";
       }
       leaf maxpoll {
         type ntp-maxpoll;
         description
           "The maximul poll interval used in this association.";
       }
       leaf prefer {
         type boolean;
         default "false";
         description
           "Whether this association is preferred.";
       }
       leaf burst {
         type boolean;
         default "false";
         description
           "Sends a series of packets instead of a single packet
           within each synchronization interval to achieve faster
           synchronization.";
       }
       leaf iburst {
         type boolean;
         default "false";
         description
           "Sends a series of packets instead of a single packet
           within the initial synchronization interval to achieve
           faster initial synchronization.";
       }
       leaf vrf {
         type string;
         description
           "The VRF instance this association binded to.";
       }
       leaf source {
         type leafref {
         path "/if:interfaces/if:interface/if:name";
        }
        description
          "The interface whose ip address this association used
          as source address.";
       }
     }


     /* Configuration data nodes */
     container ntp-cfg {
       presence
         "Enables NTP unless the ’ntp-enabled’ leaf
         (which defaults to ’true’) is set to ’false’";
       description
         "Configuration parameters for NTP.";
       leaf ntp-enabled {
         type boolean;
         default true;
         description
           "Controls whether NTP is enabled or disabled
           on this device.";
    }

    container refclock-master {
      description
        "Configuration for reference clock.";
         leaf master {
           type boolean;
           default false;
           description
             "Use its own NTP master clock to synchronize with peers
             when true.";
       }
         leaf master-stratum {
           type ntp-stratum;
           default "16";
           description
             "Use its own NTP master clock to synchronize with peers
             when true.";
      }
    }

    container authentication {
         presence
           "Enables NTP authentication when the 'auth-enabled'
           leaf is set to 'true'.";
      description
            "Configuration of authentication.";
         leaf auth-enabled {
           type boolean;
           default false;
           description
             "Controls whether NTP authentication is enabled
             or disabled on this device.";
         }
       leaf trusted-key {
       type uint32;
            description
              "The key trusted by NTP.";
         }
         list authentication-keys {
           key "key-id";
           uses authentication-key;
           description
             "List of authentication key.";
         }
    }

    container access-rules {
      description
        "Configuration of access rules.";
         list access-rule {
           key "access-mode";
           description
             "List of access rules.";
           leaf access-mode {
             type enumeration {
               enum peer {
                 description
                   "Sets the fully access authority. Both time
                   request and control query can be performed
                   on the local NTP service, and the local clock
                   can be synchronized to the remote server.";
               }
               enum server {
                 description
                   "Enables the server access and query.
                   Both time requests and control query can be
                   performed on the local NTP service, but the
                   local clock cannot be synchronized to the
                   remote server.";
               }
               enum synchronization {
                 description
                   "Enables the server to access.
                   Only time request can be performed on the
                   local NTP service.";
               }
               enum query {
                 description
                   "Sets the maximum access limitation.
                   Control query can be performed only on the
                   local NTP service.";
               }
             }
             description
               "NTP access mode.";
            }
            container acl-number {
              description
                "Configuration of acl numbers.";
              choice acl-type {
                description
                  "Type of acl.";
                case ipv4 {
                  leaf acl-number-ipv4 {
                    type uint16;
                    description "IPv4 acl number.";
                  }
                }
                case ipv6 {
                  leaf acl-number-ipv6 {
                    type uint16;
                    description "IPv6 acl number.";
                  }
                }
              }
            }
          }
    }

    container associations {
      description
        "Configuration of association.";
      container peers {
     description
              "Peer associations.";
            list peer {
              key "address vrf";
              uses association-param;
              description
                "List of peers.";
            }
      }
      container servers {
      description
              "Sever associations.";
            list server {
              key "address vrf";
              uses association-param;
              description
                "List of servers.";
            }
      }
    }

    container ntp-interfaces {
      description
        "Configuration parameters for NTP interfaces.";
      list ntp-interface {
        key "ntp-ifname";
        description
          "List of interfaces.";
        leaf ntp-ifname {
              type leafref {
              path "/if:interfaces/if:interface/if:name";
              }
              description
                "The interface name.";
        }
           container multicast-client {
             description
               "Configuration of multicast-client.";
             leaf multicast-client-address {
               type union {
              type multicast-client-v4address;
              type multicast-client-v6address;
            }
             description
               "The IP address of the multicast group to join.";
          }
           }
           container multicast-server {
             description
               "Configuration of multicast-server.";
             leaf multicast-server-address {
            type inet:ip-address;
               description
                 "The IP address to send NTP multicast packets.";
          }
             leaf multicast-server-ttl {
            type uint8;
               description
                 "Specifies the time to live (TTL) of a multicast
                 packet.";
          }
             leaf multicast-server-version {
             type ntp-version;
               description
                 "Specifies the version a multicast packet.";
          }
             leaf multicast-server-keyid {
               type leafref {
               path "/ntp:ntp-cfg/ntp:authentication/"
                     + "ntp:authentication-keys/ntp:key-id";
               }
               description
                 "Specifies the authentication key id of a
                 multicast packet.";
          }
           }
           container broadcast-client {
             description
               "Configuration of broadcast-client.";
             leaf broadcast-client-enabled {
            type boolean;
             description
               "Allows a device to receive Network Time Protocol
               (NTP) broadcast packets on an interface.";
          }
           }
           container broadcast-server {
             description
               "Configuration of broadcast-server.";
             leaf broadcast-server-version {
             type ntp-version;
               description
                 "Specifies the version of a broadcast packet.";
           }
             leaf broadcast-server-keyid {
               type leafref {
               path "/ntp:ntp-cfg/ntp:authentication/"
                       + "ntp:authentication-keys/ntp:key-id";
               }
               description
                 "Specifies the authentication key id of a
                 broadcast packet.";
           }
         }
      }
     }
     }

     /* Operational state data */
     container ntp-state {
       config "false";
       description
         "Operational state of the NTP.";

       container system-status {
         description
            "System status of NTP.";
         leaf clock-state {
           type enumeration {
              enum synchronized {
                description
                  "Indicates that the local clock has been
                  synchronized with an NTP server or
                  the reference clock.";
              }
              enum unsynchronized {
                description
                  "Indicates that the local clock has not been
                  synchronized with any NTP server.";
              }
           }
           description "Indicates the state of system clock.";
         }
         leaf clock-stratum {
             type ntp-stratum;
             description
               "Indicates the stratum of the reference clock.";
         }
         leaf clock-refid {
             type union {
                 type inet:ipv4-address;
                 type binary {
                     length "4";
                 }
                 type string {
                     length "4";
                 }
             }
             description
               "IPv4 address or first 32 bits of the MD5 hash of
               the IPv6 address or reference clock of the peer to
               which clock is synchronized.";
         }
         leaf nominal-freq {
             type decimal64 {
                 fraction-digits 4;
             }
             description
               "Indicates the nominal frequency of the
               local clock, in Hz.";
         }
         leaf actual-freq {
             type decimal64 {
                 fraction-digits 4;
             }
             description
               "Indicates the actual frequency of the
               local clock, in Hz.";
         }
         leaf clock-precision {
             type uint8;
             description
               "Precision of the clock of this system
               in Hz.(prec=2^(-n))";
         }
         leaf clock-offset {
             type decimal64 {
                 fraction-digits 4;
             }
             description
               "Offset of clock to synchronized peer,
               in milliseconds.";
         }
         leaf root-delay {
             type decimal64 {
                 fraction-digits 2;
             }
             description
               "Total delay along path to root clock,
               in milliseconds.";
         }
         leaf root-dispersion {
             type decimal64 {
                 fraction-digits 2;
             }
             description
               "Indicates the dispersion between the local clock
               and the master reference clock, in milliseconds.";
         }
         leaf peer-dispersion {
             type decimal64 {
                 fraction-digits 2;
             }
             description
               "Indicates the dispersion between the local clock
               and the peer clock, in milliseconds.";
         }
         leaf reference-time {
             type string;
             description
               "Indicates reference timestamp.";
         }
         leaf sync-state {
           type enumeration {
             enum clock-not-set {
               description
                 "Indicates the clock is not updated.";
             }
             enum freq-set-by-cfg {
               description
                 "Indicates the clock frequency is set by
                 NTP configuration.";
             }
             enum clock-set {
               description
                 "Indicates the clock is set.";
             }
             enum freq-not-determined {
               description
                 "Indicates the clock is set but the frequency
                 is not determined.";
             }
             enum clock-synchronized {
               description
                 "Indicates that the clock is synchronized.";
             }
             enum spike {
               description
                 "Indicates a time difference of more than 128
                 milliseconds is detected between NTP server
                 and client clock. The clock change will take
                 effect in XXX seconds.";
             }
           }
           description
             "Indicates the synchronization status of
             the local clock.";
         }
       }

       container associations-status {
         description
            "System status of NTP.";
         list association-status {
           key "association-source";
           description
             "List of association status.";
           leaf association-source {
             type union {
               type inet:ipv4-address;
               type inet:ipv6-address;
             }
             description
               "IPv4 or IPv6 address of the peer. If a
               nondefault VRF is configured for the peer,
               the VRF follows the address.";
           }
           leaf association-stratum {
             type ntp-stratum;
             description
               "Indicates the stratum of the reference clock.";
           }
           leaf association-refid {
             type union {
               type inet:ipv4-address;
               type binary {
                 length "4";
               }
```

```
              type string {
                length "4";
              }
            }
            description
              "Reference clock type or address for the peer.";
          }
          leaf association-reach {
            type uint8;
            description
              "Indicates the reachability of the configured
              server or peer.";
          }
          leaf association-poll {
            type uint8;
            description
              "Indicates the polling interval for current,
              in seconds.";
          }
          leaf association-now {
            type uint32;
            description
              "Indicates the time since the NTP packet was
              not received or last synchronized, in seconds.";
          }
          leaf association-offset {
            type decimal64 {
                fraction-digits 4;
            }
            description
              "Indicates the offset between the local clock
              and the superior reference clock.";
          }
          leaf association-delay {
            type decimal64 {
                fraction-digits 2;
            }
            description
              "Indicates the delay between the local clock
              and the superior reference clock.";
          }
          leaf association-dispersion {
            type decimal64 {
                fraction-digits 2;
            }
            description
              "Indicates the dispersion between the local
              clock and the superior reference clock.";
          }
          leaf association-sent {
            type uint32;
            description
              "Indicates the total number of packets this
              association sent.";
          }
          leaf association-sent-fail {
            type uint32;
            description
              "Indicates the number of times packet sending
              failed by this association.";
          }
          leaf association-received {
            type uint32;
            description
              "Indicates the total number of packets
              this association received.";
          }
          leaf association-dropped {
            type uint32;
            description
              "Indicates the number of packets
              this association dropped.";
          }
        }
      }

      container ntp-statistics {
        description
          "Packet statistics of NTP.";
        leaf packet-sent {
          type uint32;
          description
              "Indicates the total number of packets sent.";
        }
        leaf packet-sent-fail {
          type uint32;
          description
            "Indicates the number of times packet
            sending failed.";
        }
        leaf packet-received {
          type uint32;
          description
            "Indicates the total number of packets received.";
        }
        leaf packet-dropped {
          type uint32;
          description
            "Indicates the number of packets dropped.";
        }
      }
    }
  }
  //<CODE ENDS>
```

5. IANA Considerations

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

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

Registrant Contact: The NETMOD WG of the IETF.

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

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

Name: ietf-ntp

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

Prefix: ntp

Reference: RFC XXXX

6. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

7. Acknowledgments

TBD.

8. References

8.1. Normative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

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

[RFC5907] Gerstung, H., Elliott, C., and B. Haberman, "Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)", RFC 5907, June 2010.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

8.2. Informative References

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

Authors' Addresses

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

Email: eric.wu@huawei.com

Anil Kumar S N
Huawei
Kundalahalli Village, Whitefield
Bangalore, Kanataka 560037
India

Email: anil.sn@huawei.com