

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2018

G. Mirsky
X. Min
ZTE Corp.
W. Luo
Ericsson
October 20, 2017

Simple Two-way Active Measurement Protocol (STAMP) Data Model
draft-mirsky-ippm-stamp-yang-00

Abstract

This document specifies the data model for implementations of Sender and Reflector for Simple Two-way Active Measurement Protocol (STAMP) mode using YANG.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Conventions used in this document	2
1.1.1. Requirements Language	2
2. Scope, Model, and Applicability	3
2.1. Data Model Parameters	3
2.1.1. STAMP-Sender	3
2.1.2. STAMP-Reflector	4
3. Data Model	4
3.1. Tree Diagram	4
3.2. YANG Module	9
4. IANA Considerations	27
5. Security Considerations	27
6. Normative References	27
Appendix A. Acknowledgements	29
Authors' Addresses	29

1. Introduction

The Simple Two-way Active Measurement Protocol (STAMP) [I-D.mirsky-ippm-stamp] can be used to measure performance parameters of IP networks such as latency, jitter, and packet loss by sending test packets and monitoring their experience in the network. The STAMP protocol [Editor:ref to STAMP draft] in unauthenticated mode is on-wire compatible with STAMP Light, mdiscussed in Appendix I [RFC5357]. The STAMP Light is known to have many implementations though no common management framework being defined, thus leaving some aspects of test packet processing to interpretation. As one of goals of STAMP is to support these variations, this document presents their analysis; describes common STAMP and STAMP model while allowing for STAMP extensions in the future. This document defines the STAMP data model and specifies it formally using the YANG data modeling language [RFC6020].

1.1. Conventions used in this document

1.1.1. Requirements Language

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

2. Scope, Model, and Applicability

The scope of this document includes model of the STAMP as defined in [Editor:ref to STAMP draft].

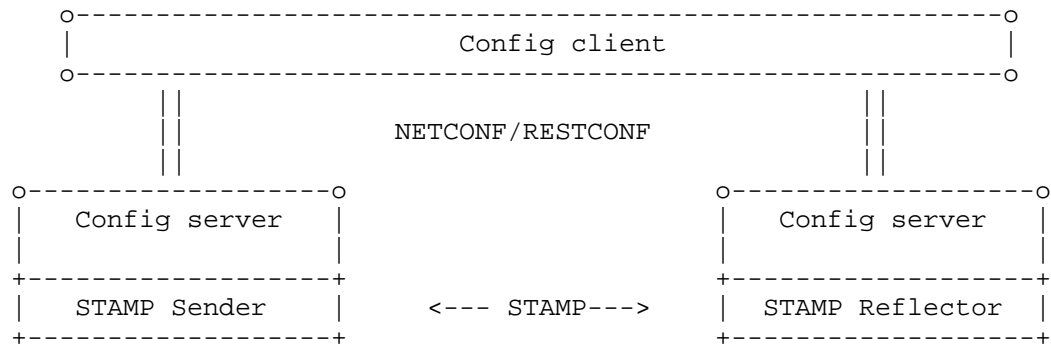


Figure 1: STAMP Reference Model

2.1. Data Model Parameters

This section describes all the parameters of the the stamp data model.

2.1.1. STAMP-Sender

The stamp-session-sender container holds items that are related to the configuration of the stamp Session-Sender logical entity.

The stamp-session-sender-state container holds information about the state of the particular STAMP test session.

RPCs stamp-sender-start and stamp-sender-stop respectively start and stop the referenced by session-id STAMP test session.

2.1.1.1. Controls for Test Session and Preformmance Metric Calculation

The data model supports several scenarios for a STAMP Sender to execute test sessions and calculate performance metrics:

The test mode in which the test packets are sent unbound in time at defined by the parameter 'interval' in the stamp-session-sender container frequency is referred as continuous mode. Performance metrics in the continuous mode are calculated at period defined by the parameter 'measurement-interval'.

The test mode that has specific number of the test packets configured for the test session in the 'number-of-packets' parameter is referred as periodic mode. The test session may be repeated by the STAMP-Sender with the same parameters. The 'repeat' parameter defines number of tests and the 'repeat-interval' - the interval between the consecutive tests. The performance metrics are calculated after each test session when the interval defined by the 'session-timeout' expires.

2.1.2. STAMP-Reflector

The stamp-session-reflector container holds items that are related to the configuration of the STAMP Session-Reflector logical entity.

The stamp-session-refl-state container holds Session-Reflector state data for the particular STAMP test session.

3. Data Model

Creating STAMP data model presents number of challenges and among them is identification of a test-session at Session-Reflector. A Session-Reflector MAY require only as little as its IP and UDP port number in received STAMP-Test packet to spawn new test session. More so, to test processing of Class-of-Service along the same route in Equal Cost Multi-Path environment Session-Sender may run STAMP test sessions concurrently using the same source IP address, source UDP port number, destination IP address, and destination UDP port number. Thus the only parameter that can be used to differentiate these test sessions would be DSCP value. The DSCP field may get re-marked along the path and without use of [RFC7750] that will go undetected, but by using five-tuple instead of four-tuple as a key we can ensure that STAMP test packets that are considered as different test sessions follow the same path even in ECMP environments.

3.1. Tree Diagram

```

module: ietf-stamp
  +--rw stamp
  |   +--rw stamp-session-sender {session-sender}?
  |   |   +--rw sender-enable?    enable
  |   |   +--rw test-session* [session-id]
  |   |       +--rw session-id                uint32
  |   |       +--rw test-session-enable?      enable
  |   |       +--rw number-of-packets?        union
  |   |       +--rw packet-padding-size?      uint32
  |   |       +--rw interval?                 uint32
  |   |       +--rw session-timeout?          uint32

```

```

+--rw measurement-interval?          uint32
+--rw repeat?                         union
+--rw repeat-interval?               uint32
+--rw dscp-value?                    inet:dscp
+--rw test-session-reflector-mode?   session-reflector-mode
+--rw sender-ip                      inet:ip-address
+--rw sender-udp-port                inet:port-number
+--rw reflector-ip                   inet:ip-address
+--rw reflector-udp-port?            inet:port-number
+--rw authentication-params! {stamp-authentication}?
|   +--rw key-chain?    kc:key-chain-ref
+--rw first-percentile?              percentile
+--rw second-percentile?             percentile
+--rw third-percentile?              percentile
+--rw stamp-session-reflector {session-reflector}?
+--rw reflector-enable?              enable
+--rw ref-wait?                      uint32
+--rw reflector-mode-state?          session-reflector-mode
+--rw test-session* [session-id]
|   +--rw session-id                 uint32
|   +--rw dscp-handling-mode?        session-dscp-mode
|   +--rw dscp-value?                inet:dscp
|   +--rw sender-ip                  inet:ip-address
|   +--rw sender-udp-port            inet:port-number
|   +--rw reflector-ip               inet:ip-address
|   +--rw reflector-udp-port?        inet:port-number
|   +--rw authentication-params! {stamp-authentication}?
|       +--rw key-chain?    kc:key-chain-ref
+--ro stamp-state
+--ro stamp-session-sender-state {session-sender}?
|   +--ro test-session-state* [session-id]
|   +--ro session-id                 uint32
|   +--ro sender-session-state?      enumeration
|   +--ro current-stats
|       +--ro start-time              yang:date-and-time
|       +--ro packet-padding-size?    uint32
|       +--ro interval?               uint32
|       +--ro duplicate-packets?      uint32
|       +--ro reordered-packets?      uint32
|       +--ro sender-ip               inet:ip-address
|       +--ro sender-udp-port          inet:port-number
|       +--ro reflector-ip             inet:ip-address
|       +--ro reflector-udp-port?      inet:port-number
|       +--ro dscp?                   inet:dscp
|       +--ro sent-packets?            uint32
|       +--ro rcv-packets?            uint32
|       +--ro sent-packets-error?      uint32
|       +--ro rcv-packets-error?      uint32

```

```

+--ro last-sent-seq?          uint32
+--ro last-rcv-seq?          uint32
+--ro two-way-delay
|   +--ro delay
|   |   +--ro min?      yang:gauge32
|   |   +--ro max?      yang:gauge32
|   |   +--ro avg?      yang:gauge32
|   +--ro delay-variation
|   |   +--ro min?      uint32
|   |   +--ro max?      uint32
|   |   +--ro avg?      uint32
+--ro one-way-delay-far-end
|   +--ro delay
|   |   +--ro min?      yang:gauge32
|   |   +--ro max?      yang:gauge32
|   |   +--ro avg?      yang:gauge32
|   +--ro delay-variation
|   |   +--ro min?      uint32
|   |   +--ro max?      uint32
|   |   +--ro avg?      uint32
+--ro one-way-delay-near-end
|   +--ro delay
|   |   +--ro min?      yang:gauge32
|   |   +--ro max?      yang:gauge32
|   |   +--ro avg?      yang:gauge32
|   +--ro delay-variation
|   |   +--ro min?      uint32
|   |   +--ro max?      uint32
|   |   +--ro avg?      uint32
+--ro low-percentile
|   +--ro delay-percentile
|   |   +--ro rtt-delay?      percentile
|   |   +--ro near-end-delay? percentile
|   |   +--ro far-end-delay?  percentile
|   +--ro delay-variation-percentile
|   |   +--ro rtt-delay-variation?      percentile
|   |   +--ro near-end-delay-variation? percentile
|   |   +--ro far-end-delay-variation?  percentile
+--ro mid-percentile
|   +--ro delay-percentile
|   |   +--ro rtt-delay?      percentile
|   |   +--ro near-end-delay? percentile
|   |   +--ro far-end-delay?  percentile
|   +--ro delay-variation-percentile
|   |   +--ro rtt-delay-variation?      percentile
|   |   +--ro near-end-delay-variation? percentile
|   |   +--ro far-end-delay-variation?  percentile
+--ro high-percentile

```

```

| | | +--ro delay-percentile
| | | | +--ro rtt-delay?           percentile
| | | | +--ro near-end-delay?     percentile
| | | | +--ro far-end-delay?      percentile
| | | +--ro delay-variation-percentile
| | | | +--ro rtt-delay-variation? percentile
| | | | +--ro near-end-delay-variation? percentile
| | | | +--ro far-end-delay-variation? percentile
+--ro two-way-loss
| | +--ro loss-count?             int32
| | +--ro loss-ratio?            percentage
| | +--ro loss-burst-max?        int32
| | +--ro loss-burst-min?        int32
| | +--ro loss-burst-count?      int32
+--ro one-way-loss-far-end
| | +--ro loss-count?            int32
| | +--ro loss-ratio?            percentage
| | +--ro loss-burst-max?        int32
| | +--ro loss-burst-min?        int32
| | +--ro loss-burst-count?      int32
+--ro one-way-loss-near-end
| | +--ro loss-count?            int32
| | +--ro loss-ratio?            percentage
| | +--ro loss-burst-max?        int32
| | +--ro loss-burst-min?        int32
| | +--ro loss-burst-count?      int32
+--ro history-stats* [id]
| | +--ro id                     uint32
| | +--ro end-time               yang:date-and-time
| | +--ro number-of-packets?     uint32
| | +--ro packet-padding-size?   uint32
| | +--ro interval?              uint32
| | +--ro duplicate-packets?     uint32
| | +--ro reordered-packets?     uint32
| | +--ro loss-packets?          uint32
| | +--ro sender-ip              inet:ip-address
| | +--ro sender-udp-port        inet:port-number
| | +--ro reflector-ip           inet:ip-address
| | +--ro reflector-udp-port?    inet:port-number
| | +--ro dscp?                  inet:dscp
| | +--ro sent-packets?          uint32
| | +--ro rcv-packets?           uint32
| | +--ro sent-packets-error?    uint32
| | +--ro rcv-packets-error?     uint32
| | +--ro last-sent-seq?         uint32
| | +--ro last-rcv-seq?         uint32
+--ro two-way-delay
| | +--ro delay

```

```
|
|
|   |   |--ro min?      yang:gauge32
|   |   +--ro max?     yang:gauge32
|   |   +--ro avg?     yang:gauge32
|   |   +--ro delay-variation
|   |       +--ro min?    uint32
|   |       +--ro max?    uint32
|   |       +--ro avg?    uint32
|   |   +--ro one-way-delay-far-end
|   |       +--ro delay
|   |           +--ro min?    yang:gauge32
|   |           +--ro max?    yang:gauge32
|   |           +--ro avg?    yang:gauge32
|   |       +--ro delay-variation
|   |           +--ro min?    uint32
|   |           +--ro max?    uint32
|   |           +--ro avg?    uint32
|   |   +--ro one-way-delay-near-end
|   |       +--ro delay
|   |           +--ro min?    yang:gauge32
|   |           +--ro max?    yang:gauge32
|   |           +--ro avg?    yang:gauge32
|   |       +--ro delay-variation
|   |           +--ro min?    uint32
|   |           +--ro max?    uint32
|   |           +--ro avg?    uint32
|--ro stamp-session-refl-state {session-reflector}?
|   |--ro reflector-light-admin-status    boolean
|   |--ro test-session-state* [session-id]
|       |--ro session-id                uint32
|       |--ro sent-packets?              uint32
|       |--ro rcv-packets?               uint32
|       |--ro sent-packets-error?        uint32
|       |--ro rcv-packets-error?         uint32
|       |--ro last-sent-seq?             uint32
|       |--ro last-rcv-seq?              uint32
|       |--ro sender-ip                  inet:ip-address
|       |--ro sender-udp-port            inet:port-number
|       |--ro reflector-ip               inet:ip-address
|       |--ro reflector-udp-port?        inet:port-number
rpcs:
+---x stamp-sender-start
|   +---w input
|       +---w session-id    uint32
+---x stamp-sender-stop
|   +---w input
|       +---w session-id    uint32
```


3.2. YANG Module

```
<CODE BEGINS> file "ietf-stamp@2017-10-20.yang"

module ietf-stamp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-stamp";
  //namespace need to be assigned by IANA
  prefix "ietf-stamp";

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-key-chain {
    prefix kc;
  }

  organization
    "IETF IPPM (IP Performance Metrics) Working Group";

  contact
    "draft-mirsky-ippm-stamp-yang@tools.ietf.org";

  description "STAMP Data Model";

  revision "2017-10-20" {
    description
      "00 version. Base STAMP specification is covered";
    reference "";
  }

  feature session-sender {
    description
      "This feature relates to the device functions as the
      STAMP Session-Sender";
  }

  feature session-reflector {
    description
      "This feature relates to the device functions as the
      STAMP Session-Reflector";
  }

  feature stamp-authentication {
    description
```

```
"STAMP authentication supported";
}

typedef enable {
    type boolean;
    description "enable";
}

typedef session-reflector-mode {
    type enumeration {
        enum stateful {
            description
            "When the Session-Reflector is stateful,
            i.e. is aware of STAMP-Test session state.";
        }
        enum stateless {
            description
            "When the Session-Reflector is stateless,
            i.e. is not aware of the state of
            STAMP-Test session.";
        }
    }
    description "State of the Session-Reflector";
}

typedef session-dscp-mode {
    type enumeration {
        enum copy-received-value {
            description
            "Use DSCP value copied from received
            STAMP test packet of the test session.";
        }
        enum use-configured-value {
            description
            "Use DSCP value configured for this
            test session on the Session-Reflector.";
        }
    }
    description
    "DSCP handling mode by Session-Reflector.";
}

typedef percentage {
    type decimal64 {
        fraction-digits 5;
    }
    description "Percentage";
}
```

```
typedef percentile {
    type decimal64 {
        fraction-digits 2;
    }
    description
    "Percentile is a measure used in statistics
    indicating the value below which a given
    percentage of observations in a group of
    observations fall.";
}

grouping maintenance-statistics {
    description "Maintenance statistics grouping";
    leaf sent-packets {
        type uint32;
        description "Packets sent";
    }
    leaf rcv-packets {
        type uint32;
        description "Packets received";
    }
    leaf sent-packets-error {
        type uint32;
        description "Packets sent error";
    }
    leaf rcv-packets-error {
        type uint32;
        description "Packets received error";
    }
    leaf last-sent-seq {
        type uint32;
        description "Last sent sequence number";
    }
    leaf last-rcv-seq {
        type uint32;
        description "Last received sequence number";
    }
}

grouping stamp-session-percentile {
    description "Percentile grouping";
    leaf first-percentile {
        type percentile;
        default 95.00;
        description
        "First percentile to report";
    }
    leaf second-percentile {
```

```
    type percentile;
    default 99.00;
    description
      "Second percentile to report";
  }
  leaf third-percentile {
    type percentile;
    default 99.90;
    description
      "Third percentile to report";
  }
}

grouping delay-statistics {
  description "Delay statistics grouping";
  container delay {
    description "Packets transmitted delay";
    leaf min {
      type yang:gauge32;
      units microseconds;
      description
        "Min of Packets transmitted delay";
    }
    leaf max {
      type yang:gauge32;
      units microseconds;
      description
        "Max of Packets transmitted delay";
    }
    leaf avg {
      type yang:gauge32;
      units microseconds;
      description
        "Avg of Packets transmitted delay";
    }
  }
}

container delay-variation {
  description
    "Packets transmitted delay variation";
  leaf min {
    type uint32;
    units microseconds;
    description
      "Min of Packets transmitted
      delay variation";
  }
  leaf max {
```

```
        type uint32;
        units microseconds;
        description
        "Max of Packets transmitted
        delay variation";
    }
    leaf avg {
        type uint32;
        units microseconds;
        description
        "Avg of Packets transmitted
        delay variation";
    }
}

grouping time-percentile-report {
    description "Delay percentile report grouping";
    container delay-percentile {
        description
        "Report round-trip, near- and far-end delay";
        leaf rtt-delay {
            type percentile;
            description
            "Percentile of round-trip delay";
        }
        leaf near-end-delay {
            type percentile;
            description
            "Percentile of near-end delay";
        }
        leaf far-end-delay {
            type percentile;
            description
            "Percentile of far-end delay";
        }
    }
}

container delay-variation-percentile {
    description
    "Report round-trip, near- and far-end delay variation";
    leaf rtt-delay-variation {
        type percentile;
        description
        "Percentile of round-trip delay-variation";
    }
    leaf near-end-delay-variation {
        type percentile;
        description
        "Percentile of near-end delay variation";
    }
}
```

```
    }
    leaf far-end-delay-variation {
        type percentile;
        description
            "Percentile of far-end delay-variation";
    }
}

grouping packet-loss-statistics {
    description
        "Grouping for Packet Loss statistics";
    leaf loss-count {
        type int32;
        description
            "Number of lost packets
            during the test interval.";
    }
    leaf loss-ratio {
        type percentage;
        description
            "Ratio of packets lost to packets
            sent during the test interval.";
    }
    leaf loss-burst-max {
        type int32;
        description
            "Maximum number of consecutively
            lost packets during the test interval.";
    }
    leaf loss-burst-min {
        type int32;
        description
            "Minimum number of consecutively
            lost packets during the test interval.";
    }
    leaf loss-burst-count {
        type int32;
        description
            "Number of occasions with packet
            loss during the test interval.";
    }
}

grouping session-parameters {
    description
        "Parameters common among
        Session-Sender and Session-Reflector";
```

```
leaf sender-ip {
  type inet:ip-address;
  mandatory true;
  description "Sender IP address";
}
leaf sender-udp-port {
  type inet:port-number {
    range "49152..65535";
  }
  mandatory true;
  description "Sender UDP port number";
}
leaf reflector-ip {
  type inet:ip-address;
  mandatory true;
  description "Reflector IP address";
}
leaf reflector-udp-port {
  type inet:port-number {
    range "862 | 49152..65535";
  }
  default 862;
  description "Reflector UDP port number";
}
}

grouping session-auth-params {
  description
    "Grouping for STAMP authentication parameters";
  container authentication-params {
    if-feature stamp-authentication;
    presence "Enables STAMP authentication";
    description
      "Parameters for STAMP Light authentication";
    leaf key-chain {
      type kc:key-chain-ref;
      description "Name of key-chain";
    }
  }
}

/* Configuration Data */
container stamp {
  description
    "Top level container for stamp configuration";

  container stamp-session-sender {
    if-feature session-sender;
  }
}
```

```
description "stamp Session-Sender container";

leaf sender-enable {
  type enable;
  default "true";
  description
    "Whether this network element is enabled to
    act as STAMP Sender";
}

list test-session {
  key "session-id";
  unique "sender-ip sender-udp-port reflector-ip"
  +" reflector-udp-port dscp-value";
  description
    "This structure is a container of test session
    managed objects";

  leaf session-id {
    type uint32;
    description "Session ID";
  }

  leaf test-session-enable {
    type enable;
    default "true";
    description
      "Whether this STAMP Test session is enabled";
  }

  leaf number-of-packets {
    type union {
      type uint32 {
        range 1..4294967294 {
          description
            "The overall number of UDP test packet
            to be transmitted by the sender for this
            test session";
        }
      }
      type enumeration {
        enum forever {
          description
            "Indicates that the test session SHALL
            be run *forever*.";
        }
      }
    }
  }
}
```



```
    default 10;
    description
      "This value determines if the STAMP-Test session is
       bound by number of test packets or not.";
  }

  leaf packet-padding-size {
    type uint32;
    default 27;
    description
      "Size of the Packet Padding. Suggested to run
       Path MTU Discovery to avoid packet fragmentation in
       IPv4 and packet blackholing in IPv6";
  }

  leaf interval {
    type uint32;
    units microseconds;
    description
      "Time interval between transmission of two
       consecutive packets in the test session in
       microseconds";
  }

  leaf session-timeout {
    when "../number-of-packets != 'forever'" {
      description
        "Test session timeout only valid if the
         test mode is periodic.";
    }
    type uint32;
    units "seconds";
    default 900;
    description
      "The timeout value for the Session-Sender to
       collect outstanding reflected packets.";
  }

  leaf measurement-interval {
    when "../number-of-packets = 'forever'" {
      description
        "Valid only when the test to run forever,
         i.e. continuously.";
    }
    type uint32;
    units "seconds";
    default 60;
    description
```

```
"Interval to calculate performance metric when
  the test mode is 'continuous'.";
}

leaf repeat {
  type union {
    type uint32 {
      range 0..4294967294;
    }
    type enumeration {
      enum forever {
        description
          "Indicates that the test session SHALL
          be repeated *forever* using the
          information in repeat-interval
          parameter, and SHALL NOT decrement
          the value.";
      }
    }
  }
  default 0;
  description
    "This value determines if the STAMP-Test session must
    be repeated. When a test session has completed, the
    repeat parameter is checked. The default value
    of 0 indicates that the session MUST NOT be repeated.
    If the repeat value is 1 through 4,294,967,294
    then the test session SHALL be repeated using the
    information in repeat-interval parameter.
    The implementation MUST decrement the value of repeat
    after determining a repeated session is expected.";
}

leaf repeat-interval {
  when "../repeat != '0'";
  type uint32;
  units seconds;
  default 0;
  description
    "This parameter determines the timing of repeated
    STAMP-Test sessions when repeat is more than 0.";
}

leaf dscp-value {
  type inet:dscp;
  default 0;
  description
    "DSCP value to be set in the test packet.";
```

```
    }

    leaf test-session-reflector-mode {
      type session-reflector-mode;
      default "stateless";
      description
        "The mode of STAMP-Reflector for the test session.";
    }

    uses session-parameters;
    uses session-auth-params;
    uses stamp-session-percentile;
  }
}

container stamp-session-reflector {
  if-feature session-reflector;
  description
    "stamp Session-Reflector container";
  leaf reflector-enable {
    type enable;
    default "true";
    description
      "Whether this network element is enabled to
      act as stamp Reflector";
  }

  leaf ref-wait {
    type uint32 {
      range 1..604800;
    }
    units seconds;
    default 900;
    description
      "REFWAIT(STAMP test session timeout in seconds),
      the default value is 900";
  }

  leaf reflector-mode-state {
    type session-reflector-mode;
    default stateless;
    description
      "The state of the mode of the stamp
      Session-Reflector";
  }

  list test-session {
    key "session-id";
```

```

        unique "sender-ip sender-udp-port reflector-ip"
        +" reflector-udp-port";
        description
        "This structure is a container of test session
        managed objects";

    leaf session-id {
        type uint32;
        description "Session ID";
    }

    leaf dscp-handling-mode {
        type session-dscp-mode;
        default copy-received-value;
        description
        "Session-Reflector handling of DSCP:
        - use value copied from received STAMP-Test packet;
        - use value explicitly configured";
    }

    leaf dscp-value {
        when "../dscp-handling-mode = 'use-configured-value'";
        type inet:dscp;
        default 0;
        description
        "DSCP value to be set in the reflected packet
        if dscp-handling-mode is set to use-configured-value.";
    }

    uses session-parameters;
    uses session-auth-params;
}
}

/* Operational state data nodes */
container stamp-state{
    config "false";
    description
    "Top level container for stamp state data";

    container stamp-session-sender-state {
        if-feature session-sender;
        description
        "Session-Sender container for state data";
        list test-session-state{
            key "session-id";
            description

```

```
"This structure is a container of test session
managed objects";

leaf session-id {
  type uint32;
  description "Session ID";
}

leaf sender-session-state {
  type enumeration {
    enum active {
      description "Test session is active";
    }
    enum ready {
      description "Test session is idle";
    }
  }
  description
  "State of the particular stamp test
  session at the sender";
}

container current-stats {
  description
  "This container contains the results for the current
  Measurement Interval in a Measurement session ";
  leaf start-time {
    type yang:date-and-time;
    mandatory true;
    description
    "The time that the current Measurement Interval started";
  }

  leaf packet-padding-size {
    type uint32;
    default 27;
    description
    "Size of the Packet Padding. Suggested to run
    Path MTU Discovery to avoid packet fragmentation
    in IPv4 and packet backholing in IPv6";
  }

  leaf interval {
    type uint32;
    units microseconds;
    description
    "Time interval between transmission of two
    consecutive packets in the test session";
  }
}
```

```
    }

    leaf duplicate-packets {
        type uint32;
        description "Duplicate packets";
    }
    leaf reordered-packets {
        type uint32;
        description "Reordered packets";
    }

    uses session-parameters;
    leaf dscp {
        type inet:dscp;
        description
            "The DSCP value that was placed in the header of
            STAMP UDP test packets by the Session-Sender.";
    }
    uses maintenance-statistics;

    container two-way-delay {
        description
            "two way delay result of the test session";
        uses delay-statistics;
    }

    container one-way-delay-far-end {
        description
            "one way delay far-end of the test session";
        uses delay-statistics;
    }

    container one-way-delay-near-end {
        description
            "one way delay near-end of the test session";
        uses delay-statistics;
    }

    container low-percentile {
        when "/stamp/stamp-session-sender/"
            +"test-session[session-id]/"
            +"first-percentile != '0.00'" {
            description
                "Only valid if the
                the first-percentile is not NULL";
        }
        description
            "Low percentile report";
    }
}
```

```
        uses time-percentile-report;
    }

    container mid-percentile {
        when "/stamp/stamp-session-sender/"
        +"test-session[session-id]/"
        +"second-percentile != '0.00'" {
            description
                "Only valid if the
                the first-percentile is not NULL";
        }
        description
            "Mid percentile report";
        uses time-percentile-report;
    }

    container high-percentile {
        when "/stamp/stamp-session-sender/"
        +"test-session[session-id]/"
        +"third-percentile != '0.00'" {
            description
                "Only valid if the
                the first-percentile is not NULL";
        }
        description
            "High percentile report";
        uses time-percentile-report;
    }

    container two-way-loss {
        description
            "two way loss count and ratio result of
            the test session";
        uses packet-loss-statistics;
    }

    container one-way-loss-far-end {
        when "/stamp/stamp-session-sender/"
        +"test-session[session-id]/"
        +"test-session-reflector-mode = 'stateful'" {
            description
                "One-way statistic is only valid if the
                session-reflector is in stateful mode.";
        }
        description
            "one way loss count and ratio far-end of
            the test session";
        uses packet-loss-statistics;
    }
}
```

```
    container one-way-loss-near-end {
      when "/stamp/stamp-session-sender/"
      +"test-session[session-id]/"
      +"test-session-reflector-mode = 'stateful'" {
        description
          "One-way statistic is only valid if the
          session-reflector is in stateful mode.";
      }
      description
        "one way loss count and ratio near-end of
        the test session";
      uses packet-loss-statistics;
    }
  }

  list history-stats {
    key id;
    description
      "This container contains the results for the history
      Measurement Interval in a Measurement session ";
    leaf id {
      type uint32;
      description
        "The identifier for the Measurement Interval
        within this session";
    }
    leaf end-time {
      type yang:date-and-time;
      mandatory true;
      description
        "The time that the Measurement Interval ended";
    }
    leaf number-of-packets {
      type uint32;
      description
        "The overall number of UDP test packets to be
        transmitted by the sender for this test session";
    }
    leaf packet-padding-size {
      type uint32;
      default 27;
      description
        "Size of the Packet Padding. Suggested to run
        Path MTU Discovery to avoid packet fragmentation
        in IPv4 and packet blackholing in IPv6";
    }
  }
```



```
    leaf interval {
      type uint32;
      units microseconds;
      description
        "Time interval between transmission of two
        consecutive packets in the test session";
    }
    leaf duplicate-packets {
      type uint32;
      description "Duplicate packets";
    }
    leaf reordered-packets {
      type uint32;
      description "Reordered packets";
    }
    leaf loss-packets {
      type uint32;
      description "Loss packets";
    }
  }

  uses session-parameters;
  leaf dscp {
    type inet:dscp;
    description
      "The DSCP value that was placed in the header of
      STAMP UDP test packets by the Session-Sender.";
  }
  uses maintenance-statistics;

  container two-way-delay{
    description
      "two way delay result of the test session";
    uses delay-statistics;
  }
  container one-way-delay-far-end{
    description
      "one way delay far end of the test session";
    uses delay-statistics;
  }
  container one-way-delay-near-end{
    description
      "one way delay near end of the test session";
    uses delay-statistics;
  }
}
}
```

```
container stamp-session-refl-state {
  if-feature session-reflector;
  description
    "stamp Session-Reflector container for
    state data";
  leaf reflector-light-admin-status {
    type boolean;
    mandatory "true";
    description
      "Whether this network element is enabled to
      act as stamp Reflector";
  }

  list test-session-state {
    key "session-id";
    description
      "This structure is a container of test session
      managed objects";

    leaf session-id {
      type uint32;
      description "Session ID";
    }

    uses maintenance-statistics;
    uses session-parameters;
  }
}

rpc stamp-sender-start {
  description
    "start the configured sender session";
  input {
    leaf session-id {
      type uint32;
      mandatory true;
      description
        "The session to be started";
    }
  }
}

rpc stamp-sender-stop {
  description
    "stop the configured sender session";
  input {
    leaf session-id {
```

```
        type uint32;
        mandatory true;
        description
            "The session to be stopped";
    }
}
```

<CODE ENDS>

4. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

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

Registrant Contact: The IPPM 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-stamp

namespace: urn:ietf:params:xml:ns:yang:ietf-stamp

prefix: stamp

reference: RFC XXXX

5. Security Considerations

The configuration, state, action data defined in this document may be accessed via the NETCONF protocol [RFC6241]. SSH [RFC6242] is mandatory secure transport that is the lowest NETCONF layer. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

6. Normative References

- [I-D.mirsky-ippm-stamp]
Mirsky, G. and G. Jun, "Simple Two-way Active Measurement Protocol", draft-mirsky-ippm-stamp-00 (work in progress), October 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC7750] Hedin, J., Mirsky, G., and S. Baillargeon, "Differentiated Service Code Point and Explicit Congestion Notification Monitoring in the Two-Way Active Measurement Protocol (TWAMP)", RFC 7750, DOI 10.17487/RFC7750, February 2016, <<https://www.rfc-editor.org/info/rfc7750>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Appendix A. Acknowledgements

Authors recognize and appreciate valuable comments provided by Adrian Pan.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Xiao Min
ZTE Corp.

Email: xiao.min2@zte.com.cn

Wei S Luo
Ericsson

Email: wei.s.luo@ericsson.com