Two-Way Active Measurement Protocol (TWAMP) Data Model
draft-cmzrjp-ippm-twamp-yang-02

Abstract

   This document specifies a data model for client and server
   implementations of the Two-Way Active Measurement Protocol (TWAMP).
   We define the TWAMP data model through Unified Modeling Language
   (UML) class diagrams and formally specify it using YANG.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] is used to
measure network performance parameters such as latency, bandwidth,
and packet loss by sending probe packets and measuring their
experience in the network. To date, TWAMP implementations do not
come with a standard management framework and, as such, configuration
depends on the various proprietary mechanisms developed by the
corresponding TWAMP vendor. This document addresses this gap by
formally specifying the TWAMP data model using YANG.

1.1. Motivation

In current TWAMP deployments, the lack of a standardized data model
limits the flexibility to dynamically instantiate TWAMP-based
measurements across equipment from different vendors. In large,
virtualized, and dynamically instantiated infrastructures where
network functions are placed according to orchestration algorithms as
discussed in [I-D.unify-nfvrg-challenges][I-D.unify-nfvrg-devops],
proprietary mechanisms for managing TWAMP measurements pose severe
limitations with respect to programmability.

Two major trends call for revisiting the standardization on TWAMP
management aspects. First, we expect that in the coming years large-
scale and multi-vendor TWAMP deployments will become the norm. From
an operations perspective, dealing with several vendor-specific TWAMP
configuration mechanisms is simply unsustainable in this context.
Second, the increasingly software-defined and virtualized nature of
network infrastructures, based on dynamic service chains [NSC] and
programmable control and management planes [RFC7426] requires a well-
defined data model for TWAMP implementations. This document defines
such a TWAMP data model and specifies it formally using the YANG data
modeling language [RFC6020].

1.2. 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 [RFC2119].

1.3. Document Organization

The rest of this document is organized as follows. Section 2
presents the scope and applicability of this document. Section 3
provides a high-level overview of the TWAMP data model. Section 4
details the configuration parameters of the data model and Section 5
specifies in YANG the TWAMP data model. Section 6 lists illustrative

examples which conform to the YANG data model specified in this
document. Appendix A elaborates these examples further.

2. Scope, Model, and Applicability

The purpose of this document is the specification of a vendor-
independent data model for TWAMP implementations.

Figure 1 illustrates a redrawn version of the TWAMP logical model
found in Section 1.2 of [RFC5357]. The figure is annotated with
pointers to the UML diagrams provided in this document and associated
with the data model of the four logical entities in a TWAMP
deployment, namely the TWAMP Control-Client, Server, Session-Sender
and Session-Reflector. As per [RFC5357], unlabeled links in Figure 1
are unspecified and may be proprietary protocols.

```
    [Fig. 3]                                [Fig. 4]
+----------------+                        +--------+
| Control-Client |  <-- TWAMP-Control -->  | Server |
+----------------+                        +--------+
       ^                                       ^
       |                                       |
       V                                       V
+----------------+                    +-------------------+
| Session-Sender |  <-- TWAMP-Test --> | Session-Reflector |
+----------------+                    +-------------------+
    [Fig. 5]                                [Fig. 6]
```
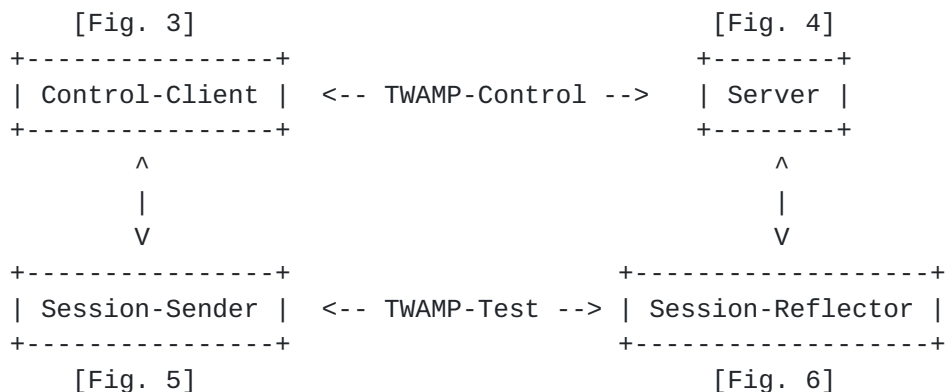
Figure 1: Annotated TWAMP logical model

As per [RFC5357], a TWAMP implementation may follow a simplified
logical model, in which the same node acts both as the Control-Client
and Session-Sender, while another node acts at the same time as the
TWAMP Server and Session-Reflector. Figure 2 illustrates this
simplified logical model and indicates the interaction between the
TWAMP configuration client and server using, for instance, NETCONF
[RFC6241] or RESTCONF [I-D.ietf-netconf-restconf]. Note, however,
that the specific protocol used to communicate the TWAMP
configuration parameters specified herein is outside the scope of
this document. Appendix B considers TWAMP operational commands,
which are also outside the scope of this document.

```
o------------------o                    o------------------o
|   Config client  |                    |   Config client  |
o------------------o                    o------------------o
       ||                                      ||
 NETCONF || RESTCONF                    NETCONF || RESTCONF
       ||                                      ||
o------------------o                    o------------------o
|   Config server  |                    |   Config server  |
|    [Fig. 3, 5]   |                    |    [Fig. 4, 6]   |
+------------------+                    +------------------+
|   Control-Client | <-- TWAMP-Control --> |     Server       |
|                  |                    |                  |
|   Session-Sender | <-- TWAMP-Test -->  | Session-Reflector |
+------------------+                    +------------------+
```
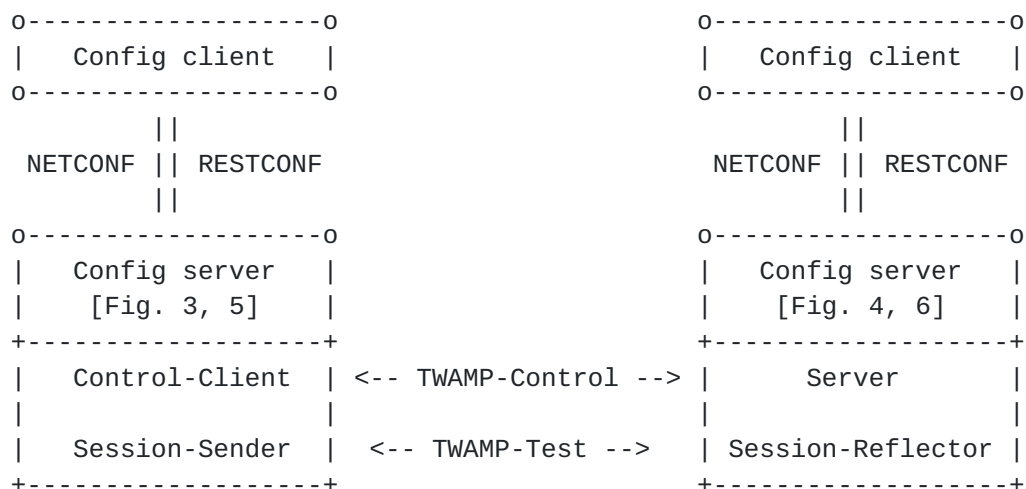
                Figure 2: Simplified TWAMP model and protocols

3. Data Model Overview

   A TWAMP data model includes four categories of configuration items.
   Global configuration items relate to parameters that are set on a per
   device level. For example, the administrative status of the device
   with respect to whether it allows TWAMP sessions and, if so, in what
   capacity (e.g. Control-Client, Server or both), are typical
   instances of global configuration items. A second category includes
   attributes that can be configured on a per control connection basis,
   such as the Server IP address. A third category includes attributes
   related to per test session attributes, for instance setting
   different values in the Differentiated Services Code Point (DSCP)
   field. Finally, the data model could include attributes that relate
   to the operational state of the TWAMP implementation.

   As we describe the TWAMP data model in the remaining sections of this
   document, readers should keep in mind the functional entity grouping
   illustrated in Figure 1.

3.1. Control-Client

   A TWAMP Control-Client has an administrative status field set at the
   device level that indicates whether the node is enabled to function
   as such.

   Each TWAMP Control-Client is associated with zero or more TWAMP
   control connections. The main configuration parameters of each
   control connection are:

   o A name which can be used to uniquely identify at the Control-
      Client a particular control connection. This name is necessary

for programmability reasons because at the time of creation of a
TWAMP control connection not all IP and TCP port number
information needed to uniquely identify the connection is
available.

o The IP address of the interface the Control-Client will use for
   connections

o The IP address of the remote Server

o Authentication and Encryption attributes such as KeyID, Token and
   the Client Initialization Vector (Client-IV) [RFC4656].

Each TWAMP control connection, in turn, is associated with zero or
more test sessions. For each test session we note the following
configuration items:

o The test session name that uniquely identifies a particular test
   session at the Control-Client and Session-Sender. Similarly to
   the control connections above, this unique test session name is
   needed because at the time of creation of a test session, for
   example, the source UDP port number is not known to uniquely
   identify the test session.

o The IP address and UDP port number of the Session-Sender of the
   path under test by TWAMP

o The IP address and UDP port number of the Session-Reflector of
   said path

o Information pertaining to the test packet stream, such as the test
   starting time or whether the test should be repeated.

3.2. Server

Each TWAMP Server has an administrative status field set at the
device level to indicate whether the node is enabled to function as a
TWAMP Server.

Each TWAMP Server is associated with zero or more control
connections. Each control connection is uniquely identified by the
4-tuple {Control-Client IP address, Control-Client TCP port number,
Server IP address, Server TCP port}. Control connection configuration
items on a TWAMP Server are read-only.

3.3. Session-Sender

   There is one TWAMP Session-Sender instance for each test session that
   is initiated from the sending device. Primary configuration fields
   include:

   o The test session name that MUST be identical with the
      corresponding test session name on the TWAMP Control-Client
      (Section 3.1)

   o The control connection name, which along with the test session
      name uniquely identify the TWAMP Session-Sender instance

   o Information pertaining to the test packet stream, such as, for
      example, the number of test packets and the packet distribution to
      be employed.

3.4. Session-Reflector

   Each TWAMP Session-Reflector is associated with zero or more test
   sessions. For each test session, the REFWAIT parameter (Section 4.2
   of [RFC5357] can be configured. Read-only access to other data model
   parameters, such as the Sender IP address is foreseen. Each test
   session can be uniquely identified by the 4-tuple mentioned in
   Section 3.2.

4. Data Model Parameters

   This section defines the TWAMP data model using UML and describes all
   associated parameters.

4.1. Control-Client

   The twamp-client container (see Figure 3) holds items that are
   related to the configuration of the TWAMP Control-Client logical
   entity. These are divided up into items that are associated with the
   configuration of the Control-Client as a whole (e.g. client-admin-
   state) and items that are associated with individual control
   connections initiated by the Control-Client entity (twamp-client-
   ctrl-connection).

```
   +--------------------+
   | twamp-client       |
   +--------------------+                        1..* +-----------------------+
   | client-admin-state |<>---------------------| mode-preference-chain |
   |                    |                        +-----------------------+
   |                    | 1..* +------------+    | priority              |
   |                    |<>-----| key-chain  |    | mode                  |
   +--------------------+       +------------+    +-----------------------+
            ^                   | key-id     |
            V                   | secret-key |
            |                   +------------+
            | 0..*
   +-----------------------------+
   | twamp-client-ctrl-connection |
   +-----------------------------+
   | ctrl-connection-name        |
   | client-ip                   |
   | server-ip                   |
   | server-tcp-port             |      0..* +-------------------------+
   | dscp                        |<>-------| twamp-session-request   |
   | key-id                      |         +-------------------------+
   | max-count                   |         | test-session-name       |
   | client-tcp-port        {ro} |         | sender-ip               |
   | server-start-time      {ro} |         | sender-udp-port         |
   | ctrl-connection-state  {ro} |         | reflector-ip            |
   | selected-mode          {ro} |         | reflector-udp-port      |
   | token                  {ro} |         | timeout                 |
   | client-iv              {ro} |         | padding-length          |
   +-----------------------------+         | dscp                    |
                                           | start-time              |
                   +-------------+ 1       | repeat                  |
                   | pm-reg-list |------<>| repeat-interval         |
                   +-------------+         | test-session-state {ro} |
                   | pm-index    |         | sid                {ro} |
                   +-------------+         +-------------------------+
```

              Figure 3: TWAMP Control-Client UML class diagram

   The twamp-client container includes an administrative parameter
   (client-admin-state) that controls whether the device is allowed to
   initiate TWAMP control sessions.

   The twamp-client container holds a list (mode-preference-chain) which
   specifies the preferred Mode values according to their preferred
   order of use, including the authentication and encryption Modes.
   Specifically, mode-preference-chain lists each priority (expressed as
   a 16-bit unsigned integer, where zero is the highest priority and
   subsequent values monotonically increasing) with their corresponding

mode (expressed as a 32-bit Hexadecimal value). Depending on the
Modes available in the Server Greeting, the Control-Client MUST
choose the highest priority Mode from the configured mode-preference-
chain list. Note that the list of preferred Modes may set bit
position combinations when necessary, such as when referring to the
extended TWAMP features in [RFC5618], [RFC5938], and [RFC6038]. If
the Control-Client cannot determine an acceptable Mode, it MUST
respond with zero Mode bits set in the Set-up Response message,
indicating it will not continue with the control connection.

In addition, the twamp-client container holds a list named key-chain
which relates KeyIDs with the respective secret keys. Both the
Server and the Control-Client use the same mappings from KeyIDs to
shared secrets (key-id and secret-key in Figure 3, respectively).
The Server, being prepared to conduct sessions with more than one
Control-Client, uses KeyIDs to choose the appropriate secret-key; a
Control-Client would typically have different secret keys for
different Servers. The secret-key is the shared secret, an octet
string of arbitrary length whose interpretation as a text string is
unspecified. The key-id and secret-key encoding should follow
Section 9.4 of [RFC6020]. The derived key length (dkLen in
[RFC2898]) MUST be 128-bits for the AES Session-key used for
encryption and a 256-bit HMAC-SHA1 Session-key used for
authentication (see Section 6.10 of [RFC4656]).

Each twamp-client container also holds a list of twamp-client-ctrl-
connection, where each item in the list describes a TWAMP control
connection that will be initiated by this Control-Client. There
SHALL be one instance of twamp-client-ctrl-connection per TWAMP-
Control (TCP) connection that is to be initiated from this device.

The configuration items for twamp-client-ctrl-connection are:

ctrl-connection-name
        A unique name used as a key to identify this individual TWAMP
        control connection on the Control-Client device.

client-ip
        The IP address of the local Control-Client device, to be
        placed in the source IP address field of the IP header in
        TWAMP-Control (TCP) packets belonging to this control
        connection. If not configured, the device SHALL choose its
        own source IP address.

server-ip
        The IP address belonging to the remote Server device, which
        the TWAMP-Control connection will be initiated to. This item
        is mandatory.

server-tcp-port

> This parameter defines the TCP port number that is to be used by this outgoing TWAMP-Control connection. Typically, this is the well-known TWAMP port number (862) as per [RFC5357]. However, there are known realizations of TWAMP in the field that were implemented before this well-known port number was allocated. These early implementations allowed the port number to be configured. This parameter is therefore provided for backward compatibility reasons. The default value is 862.

dscp   The DSCP value to be placed in the TCP header of TWAMP-Control packets generated by this Control-Client. The default value is 0.

key-id

> The key-id value that is selected for this TWAMP-Control connection.

max-count

> If an attacking system sets the maximum value in Count (2**32), then the system under attack would stall for a significant period of time while it attempts to generate keys. Therefore, TWAMP-compliant systems SHOULD have a configuration control to limit the maximum Count value. The default max-count value SHOULD be 32768.

The following twamp-client-ctrl-connection parameters are read-only:

client-tcp-port

> The source TCP port number used in the TWAMP-Control packets belonging to this control connection.

server-start-time

> The Start-Time advertised by the Server in the Server-Start message ([RFC4656], Section 3.1). This is a timestamp representing the time when the current instantiation of the Server started operating.

ctrl-connection-state

> The TWAMP-Control connection state can be either active or idle.

selected-mode

> The TWAMP Mode that the Control-Client has chosen for this control connection as set in the Mode field of the Set-Up-Response message ([RFC4656], Section 3.1).

token   This parameter holds the 64 octets containing the
         concatenation of a 16-octet challenge, a 16-octet AES
         Session-key used for encryption, and a 32-octet HMAC-SHA1
         Session-key used for authentication. AES Session-key and
         HMAC Session-key are generated randomly by the Control-
         Client. AES Session-key and HMAC Session-key MUST be
         generated with sufficient entropy not to reduce the security
         of the underlying cipher [RFC4086]. The token itself is
         encrypted using the AES (Advanced Encryption Standard) in
         Cipher Block Chaining (CBC). Encryption MUST be performed
         using an Initialization Vector (IV) of zero and a key derived
         from the shared secret associated with KeyID. Challenge is
         the same as transmitted by the Server (Section 4.2) in the
         clear; see also the last paragraph of Section 6 in [RFC4656].

client-iv
         The Control-Client Initialization Vector (Client-IV) is
         generated randomly by the Control-Client. Client-IV merely
         needs to be unique (i.e., it MUST never be repeated for
         different sessions using the same secret key; a simple way to
         achieve that without the use of cumbersome state is to
         generate the Client-IV values using a cryptographically
         secure pseudo-random number source.

Each twamp-client-ctrl-connection holds a list of twamp-session-
request. twamp-session-request holds information associated with the
Control-Client for this test session. This includes information that
is associated with the Request-TW-Session/Accept-Session message
exchange (see Section 3.5 of [RFC5357]). The Control-Client is also
responsible for scheduling and results collection for TWAMP-Test
sessions, so twamp-session-request will also hold information related
these actions (e.g. pm-index, repeat-interval).

There SHALL be one instance of twamp-session-request for each TWAMP-
Test session that is to be negotiated by this TWAMP-Control
connection via a Request-TW-Session/Accept-Session exchange.

The configuration items for twamp-session-request are:

test-session-name
         A unique name for this test session to be used for
         identification of this TWAMP-Test session on the Control-
         Client.

sender-ip
         The IP address of the Session-Sender device, which is to be
         placed in the source IP address field of the IP header in
         TWAMP-Test (UDP) packets belonging to this test session.

           This value will be used to populate the sender address field
           of the Request-TW-Session message. If not configured, the
           device SHALL choose its own source IP address.

   sender-udp-port
           The UDP port number that is to be used by the Session-Sender
           for this TWAMP-Test session. A value of zero indicates that
           the Control-Client SHALL auto-allocate a UDP port number for
           this TWAMP-Test session. The configured (or auto-allocated)
           value is advertized in the Sender Port field of the Request-
           TW-session message (see also Section 3.5 of [RFC5357]). Note
           that in the scenario where a device auto-allocates a UDP port
           number for a session, and the repeat parameter for that
           session indicates that it should be repeated, the device is
           free to auto-allocate a different UDP port number when it
           negotiates the next (repeated) iteration of this session.

   reflector-ip
           The IP address belonging to the remote Session-Reflector
           device to which the TWAMP-Test session will be initiated.
           This value will be used to populate the receiver address
           field of the Request-TW-Session message. This item is
           mandatory.

   reflector-udp-port
           This parameter defines the UDP port number that will be used
           by the Session-Reflector for this TWAMP-Test session. This
           value will be placed in the Receiver Port field of the
           Request-TW-Session message. If this value is not set, the
           device SHALL use the same port number as defined in the
           server-tcp-port parameter of this twamp-session-request's
           parent twamp-client-ctrl-connection.

   timeout The length of time (in seconds) that the Session-Reflector
           should continue to respond to packets belonging to this
           TWAMP-Test session after a Stop-Sessions TWAMP-Control
           message has been received ([RFC5357], Section 3.8). This
           value will be placed in the Timeout field of the Request-TW-
           Session message. The default value is 2 seconds.

   padding-length
           The number of bytes of padding that will be added to the
           TWAMP-Test (UDP) packets generated by the Session-Sender.
           This value will be placed in the Padding Length field of the
           Request-TW-Session message ([RFC4656], Section 3.5).

   dscp   The DSCP value to be placed in the UDP header of TWAMP-Test
           packets generated by the Session-Sender, and in the UDP

header of the TWAMP-Test response packets generated by the
Session-Reflector for this test session. This value will be
placed in the Type-P Descriptor field of the Request-TW-
Session message ([RFC5357]).

start-time
        Time when the session is to be started (but not before the
        Start-Sessions command is issued). This value is placed in
        the Start Time field of the Request-TW-Session message. The
        default value of 0 indicates that the session will be started
        as soon as the Start-Sessions message is received.

repeat and repeat-interval
        These two values together are used to determine if the TWAMP-
        Test session is to be run repeatedly. Once a test session
        has completed, the repeat parameter is checked. If the value
        indicates that this test session is to run again, then the
        parent TWAMP-Control connection for this test session is
        restarted - and negotiates a new instance of this TWAMP-Test
        session. This may occur immediately after the test session
        completes (if the repeat-interval is set to 0). Otherwise,
        the Control-Client will wait for the number of minutes
        specified in the repeat-interval parameter before negotiating
        the new instance of this TWAMP-Test session. The default
        value of repeat is 0, indicating that once the session has
        completed, it will not be renegotiated and restarted.

pm-reg-list
        A list of one or more Performance Metric Registry Index
        values (see [I-D.ietf-ippm-metric-registry], which
        communicate packet stream characteristics and one or more
        metrics to be measured. All members of the pm-reg-list MUST
        have the same stream characteristics, such that they combine
        to specify all metrics that shall be measured on a single
        stream.

pm-index
        One or more Numerical index values of a Registered Metric in
        the Performance Metric Registry
        [I-D.ietf-ippm-metric-registry] comprise the pm-reg-list.
        Output statistics are specified in the corresponding Registry
        entry.

The following twamp-session-request parameters are read-only:

test-session-state
        The TWAMP-Test session state can be either accepted or
        indicate the respective error code.

   sid   The SID allocated by the Server for this TWAMP-Test session,
           and communicated back to the Control-Client in the SID field
           of the Accept-Session message; see Section 4.3 of [RFC6038].

4.2. Server

   The twamp-server container (see Figure 4) holds items that are
   related to the configuration of the TWAMP Server logical entity
   (recall Figure 1).

```
+----------------- -+
| twamp-server      |
+-------------------+
| server-admin-state |   1..* +------------+
| server-tcp-port    |<>------| key-chain  |
| servwait           |        +------------+
| dscp               |        | key-id      |
| count              |        | secret-key |
| max-count          |        +------------+
| modes              |
|                    |    0..* +---------------------------------+
|                    |<>------| twamp-server-ctrl-connection    |
+-------------------+         +---------------------------------+
                              | client-ip                 {ro} |
                              | client-tcp-port           {ro} |
                              | server-ip                 {ro} |
                              | server-tcp-port           {ro} |
                              | server-ctrl-connection-state {ro} |
                              | dscp                      {ro} |
                              | selected-mode             {ro} |
                              | key-id                    {ro} |
                              | count                     {ro} |
                              | max-count                 {ro} |
                              | salt                      {ro} |
                              | server-iv                 {ro} |
                              | challenge                 {ro} |
                              +---------------------------------+
```
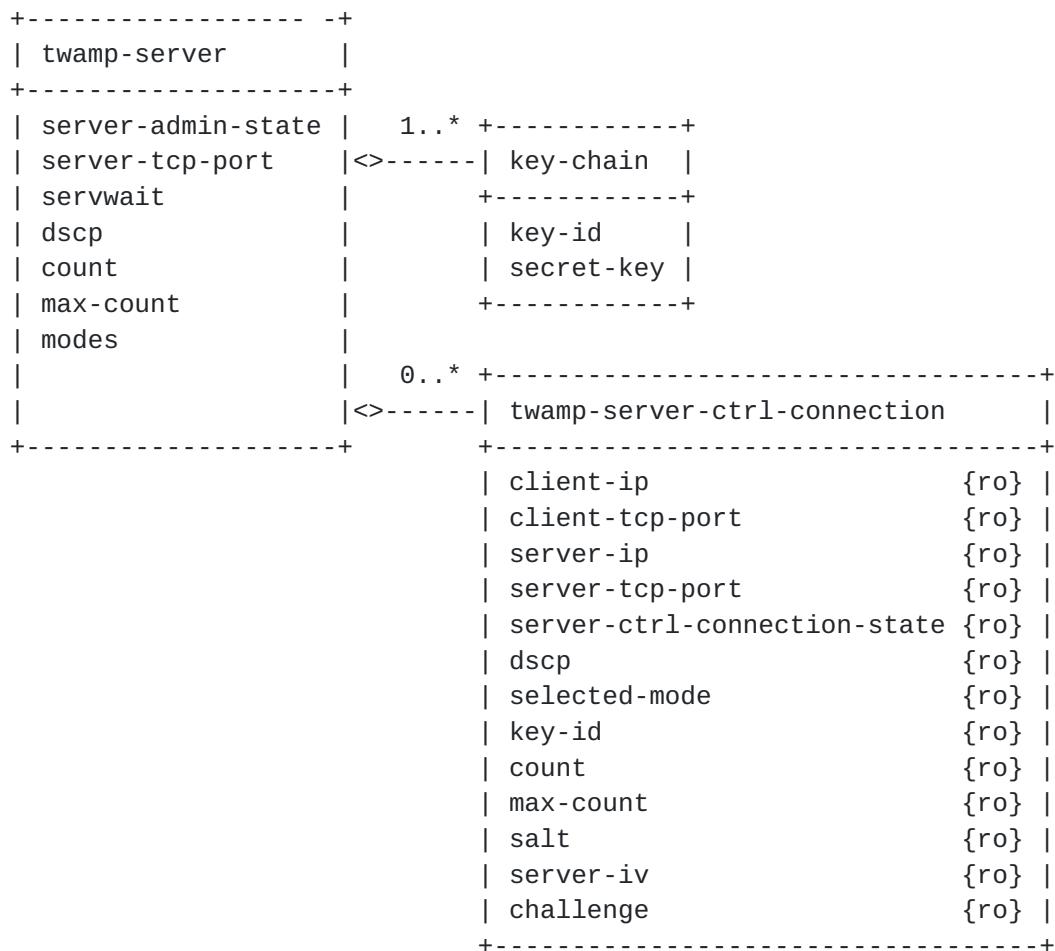
               Figure 4: TWAMP Server UML class diagram

   A device operating in the Server role cannot configure attributes on
   a per TWAMP-Control connection basis, as it has no foreknowledge of
   what incoming TWAMP-Control connections it will receive. As such,
   any parameter that the Server might want to apply to an incoming
   control connection must be configured at the overall Server level,
   and will then be applied to all incoming TWAMP-Control connections.

Each twamp-server container holds a list named key-chain which
relates KeyIDs with the respective secret keys. As mentioned in
Section 4.1, both the Server and the Control-Client use the same
mappings from KeyIDs to shared secrets. The Server, being prepared
to conduct sessions with more than one Control-Client, uses KeyIDs to
choose the appropriate secret-key; a Control-Client would typically
have different secret keys for different Servers. key-id tells the
Server which shared-secret the Control-Client wishes to use for
authentication or encryption.

Each incoming control connection that is active on the Server will be
represented by an instance of a twamp-server-ctrl-connection object.
All items in the twamp-server-ctrl-connection object are read-only,
as we explain later in this section.

The twamp-server container items are as follows:

server-admin-state
        This administrative parameter controls whether the device is
        allowed to operate as a TWAMP Server. As defined in
        [RFC5357] the roles of Server and Session-Reflector can be
        played by the same host; recall Figure 2. For a host
        operating in this manner, this parameter controls whether the
        device is allowed to respond to TWAMP control sessions.

server-tcp-port
        This parameter defines the well known TCP port number that is
        used by TWAMP-Control. The Server will listen on this port
        number for incoming TWAMP-Control connections. Although this
        is defined as a fixed value (862) in [RFC5357], there are
        several realizations of TWAMP in the field that were
        implemented before this well-known port number was allocated.
        These early implementations allowed the port number to be
        configured. This parameter is therefore provided for
        backward compatibility reasons. The default value is 862.

servwait
        TWAMP-Control (TCP) session timeout, in seconds (([RFC5357],
        Section 3.1)).

dscp   The DSCP value to be placed in the IP header of TWAMP-Control
        (TCP) packets generated by the Server. Section 3.1 of
        [RFC5357] specifies that the server SHOULD use the DSCP value
        from the Control-Client's TCP SYN. However, for practical
        purposes TWAMP will typically be implemented using a general
        purpose TCP stack provided by the underlying operating
        system, and such a stack may not provide this information to
        the user. Consequently, it is not always possible to

implement the behavior described in [RFC5357] in an OS-
portable version of TWAMP. The default behavior if this item
is not set is to use the DSCP value from the Control-Client's
TCP SYN, as per Section 3.1 of [RFC5357].

count   Parameter used in deriving a key from a shared secret as
described in Section 3.1 of [RFC4656], and are communicated
to the Control-Client as part of the Server Greeting message.
count MUST be a power of 2. count MUST be at least 1024.
count SHOULD be increased as more computing power becomes
common.

max-count
If an attacking system sets the maximum value in count
(2**32), then the system under attack would stall for a
significant period of time while it attempts to generate
keys. Therefore, TWAMP-compliant systems SHOULD have a
configuration control to limit the maximum count value. The
default max-count value SHOULD be 32768.

modes
The bit mask of TWAMP Modes this Server instance is willing
to support; see IANA TWAMP Modes Registry. Each bit position
set represents a mode; see TWAMP-Modes at
http://www.iana.org/assignments/twamp-parameters/twamp-
parameters.xhtml. Note: Modes requiring Authentication or
Encryption MUST include the related attributes.

There SHALL be one instance of twamp-server-ctrl-connection per
incoming TWAMP-Control (TCP) connection that is received and active
on the Server device. All items in the twamp-server-ctrl-connection
are read-only. Each instance of twamp-server-ctrl-connection uses
the following 4-tuple as its unique key: client-ip, client-tcp-port,
server-ip, server-tcp-port.

The twamp-server-ctrl-connection container items are all read-only:

client-ip
The IP address on the remote Control-Client device, which is
the source IP address used in the TWAMP-Control (TCP) packets
belonging to this control connection.

client-tcp-port
The source TCP port number used in the TWAMP-Control (TCP)
packets belonging to this control connection.

server-ip

          The IP address of the local Server device, which is the
          destination IP address used in the TWAMP-Control (TCP)
          packets belonging to this control connection.

   server-tcp-port
          The destination TCP port number used in the TWAMP-Control
          (TCP) packets belonging to this control connection. This
          will usually be the same value as the server-tcp-port
          configured under twamp-server. However, in the event that
          the user re-configured twamp-server:server-tcp-port after
          this control connection was initiated, this value will
          indicate the server-tcp-port that is actually in use for this
          control connection.

   server-ctrl-connection-state
          The Server TWAMP-Control connection state can be active or
          SERVWAIT.

   dscp
          The DSCP value used in the IP header of the TWAMP-Control
          (TCP) packets sent by the Server for this control connection.
          This will usually be the same value as is configured in the
          dscp parameter under the twamp-server container. However, in
          the event that the user re-configures twamp-server:dscp after
          this control connection is already in progress, this read-
          only value will show the actual dscp value in use by this
          TWAMP-Control connection.

   selected-mode
          The Mode that was chosen for this TWAMP-Control connection as
          set in the Mode field of the Set-Up-Response message.

   key-id
          The KeyID value that is in use by this TWAMP-Control
          connection. The Control-Client selects the key-id for the
          control connection.

   count
          The count value that is in use by this TWAMP-Control
          connection. This will usually be the same value as is
          configured under twamp-server. However, in the event that
          the user re-configured twamp-server:count after this control
          connection is already in progress, this read-only value will
          show the actual count that is in use for this TWAMP-Control
          connection.

   max-count

The max-count value that is in use by this TWAMP-Control
connection. This will usually be the same value as is
configured under twamp-server. However, in the event that
the user re-configured twamp-server:max-count after this
control connection is already in progress, this read-only
value will show the actual max-count that is in use for this
control connection.

salt   A parameter used in deriving a key from a shared secret as
described in Section 3.1 of [RFC4656]. Salt MUST be
generated pseudo-randomly (independently of anything else in
the RFC) and is communicated to the Control-Client as part of
the Server Greeting message.

server-iv
The Server Initialization Vector (IV) is generated randomly
by the Server.

challenge
A random sequence of octets generated by the Server. As
described in Section 4.1 challenge is used by the Control-
Client to prove possession of a shared secret.

4.3. Session-Sender

The twamp-session-sender container, illustrated in Figure 5, holds
items that are related to the configuration of the TWAMP Session-
Sender logical entity.

The twamp-session-sender container includes an administrative
parameter (session-sender-admin-state) that controls whether the
device is allowed to initiate TWAMP test sessions.

There is one instance of twamp-sender-test-session for each TWAMP-
Test session for which packets are being sent.

```
    +---------------------------+
    | twamp-session-sender      |
    +---------------------------+  0..* +---------------------------+
    | session-sender-admin-state |<>-----| twamp-sender-test-session |
    +---------------------------+        +---------------------------+
                                         | test-session-name         |
                                         | ctrl-connection-name {ro} |
                                         | fill-mode                 |
                                         | number-of-packets         |
                                         | sender-session-state {ro} |
                                         | sent-packets         {ro} |
                                         | rcv-packets          {ro} |
                                         | last-sent-seq        {ro} |
                                         | last-rcv-seq         {ro} |
                                         +---------------------------+
                                                      ^
                                                      V
                                                      | 1
                                          +--------------------+
                                          | packet-distribution |
                                          +--------------------+
                                          | periodic /  poisson |
                                          +--------------------+
                                              |          |
                             +------------------------+  |
                             | periodic-interval       |  |
                             | periodic-interval-units |  |
                             +------------------------+  |
                                          +------------------------+
                                          | lambda                 |
                                          | lambda-units           |
                                          | max-interval           |
                                          | truncation-point-units |
                                          +------------------------+
```

            Figure 5: TWAMP Session-Sender UML class diagram

   The twamp-sender-test-session container items are:

   test-session-name
            A unique name for this TWAMP-Test session to be used for
            identifying this test session by the Session-Sender logical
            entity.

   ctrl-connection-name
            The name of the parent TWAMP-Control connection that is
            responsible for negotiating this TWAMP-Test session.

fill-mode
        Indicates whether the padding added to the TWAMP-Test (UDP)
        packets will contain pseudo-random numbers, or whether it
        should consist of all zeroes, as per Section 4.2.1 of
        [RFC5357].

number-of-packets
        The overall number of TWAMP-Test (UDP) packets to be
        transmitted by the Session-Sender for this test session.

packet-distribution
        Defines whether TWAMP-Test (UDP) packets are to be
        transmitted with a fixed interval between them, or whether a
        Poisson distribution is to be used.

periodic-interval and periodic-interval-units
        If packet-distribution is set to periodic, these two values
        are used together to determine the period to wait between the
        first bits of TWAMP-Test (UDP) packet transmissions for this
        test session. periodic-interval-units is one of seconds,
        milliseconds, microseconds, nanoseconds; see [RFC3432].

lambda and lambda-units
        If packet-distribution is Poisson, the lambda parameter
        determines the corresponding average rate of packet
        transmission. lambda-units defines the units of lambda in
        reciprocal seconds; see [RFC3432].

max-interval
        If packet-distribution is Poisson, then this parameter keeps
        a stream active by setting a maximum time between packet
        transmissions.

truncation-point-units
        One of seconds, milliseconds, microseconds, nanoseconds.

The following twamp-sender-test-session parameters are read-only:

sender-session-state
        This read-only item can be either Active or Idle.

sent-packets
        The number of TWAMP-Test (UDP) packets belonging to this
        session that have been transmitted by the Session-Sender.

rcv-packets
        The number of TWAMP-Test (UDP) packets belonging to this
        session that have been received from the Session-Reflector.

The round trip loss for a test session can be calculated as
sent-packets - rcv-packets.

last-sent-seq
        The value in the sequence number field of the last TWAMP-Test
        (UDP) packet transmitted for this test session. Sequence
        numbers start from zero, so this should always be one less
        than the sent-packets value.

last-rcv-seq
        The value in the sequence number field of the last TWAMP-Test
        (UDP) packet received for this test session. In the case of
        packet loss in the Session-Sender to Session-Reflector
        direction, this value minus the last-sent-seq will quantify
        the number of packets that were lost in the Session-Sender to
        Session-Reflector direction.

4.4. Session-Reflector

The twamp-session-reflector container, illustrated in Figure 6, holds
items that are related to the configuration of the TWAMP Session-
Reflector logical entity.

A device operating in the Session-Reflector role cannot configure
attributes on a per-session basis, as it has no foreknowledge of what
incoming sessions it will receive. As such, any parameter that the
Session-Reflector might want to apply to an incoming TWAMP-Test
session must be configured at the overall Session-Reflector level,
and will then be applied to all incoming sessions.

The twamp-session-sender container includes an administrative
parameter (session-reflector-admin-state) that controls whether the
device is allowed to respond to incoming TWAMP test sessions. Each
incoming TWAMP-Test session that is active on the Session-Reflector
will be represented by an instance of a twamp-reflector-test-session
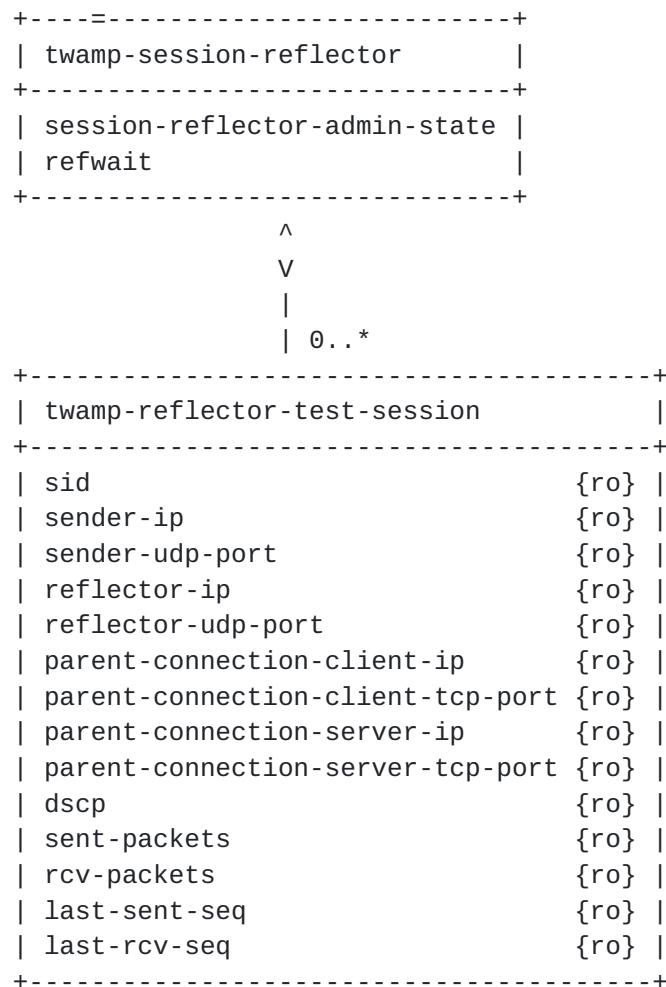object. All items in the twamp-reflector-test-session object are
read-only.

```
+----=-------------------------+
| twamp-session-reflector      |
+------------------------------+
| session-reflector-admin-state |
| refwait                      |
+------------------------------+
                ^
                V
                |
                | 0..*
+----------------------------------------+
| twamp-reflector-test-session           |
+----------------------------------------+
| sid                            {ro} |
| sender-ip                      {ro} |
| sender-udp-port                {ro} |
| reflector-ip                   {ro} |
| reflector-udp-port             {ro} |
| parent-connection-client-ip    {ro} |
| parent-connection-client-tcp-port {ro} |
| parent-connection-server-ip    {ro} |
| parent-connection-server-tcp-port {ro} |
| dscp                           {ro} |
| sent-packets                   {ro} |
| rcv-packets                    {ro} |
| last-sent-seq                  {ro} |
| last-rcv-seq                   {ro} |
+----------------------------------------+
```

          Figure 6: TWAMP Session-Reflector UML class diagram

   The twamp-session-reflector configuration items are:

   refwait
           The Session-Reflector MAY discontinue any session that has
           been started when no packet associated with that session has
           been received for REFWAIT seconds. The default value of
           REFWAIT SHALL be 900 seconds, and this waiting time MAY be
           configurable. This timeout allows a Session-Reflector to
           free up resources in case of failure.

   Instances of twamp-reflector-test-session are indexed by a session
   identifier (sid). This value is auto-allocated by the Server as test
   session requests are received, and communicated back to the Control-
   Client in the SID field of the Accept-Session message; see
   Section 4.3 of [RFC6038].

When attempting to retrieve operational data for active test sessions from a Session-Reflector device, the user will not know what sessions are currently active on that device, or what SIDs have been auto-allocated for these test sessions. If the user has network access to the Control-Client device, then it is possible to read the data for this session under twamp-client:twamp-client-ctrl-connection:twamp-session-request:sid and obtain the SID (see Figure 3). The user may then use this SID value as an index to retrieve an individual twamp-session-reflector:twamp-reflector-test-session instance on the Session-Reflector device.

If the user has no network access to the Control-Client device, then the only option is to retrieve all twamp-reflector-test-session instances from the Session-Reflector device. This could be problematic if a large number of test sessions are currently active on that device.

Each Session-Reflector TWAMP-Test session contains the following 4-tuple: {parent-connection-client-ip, parent-connection-client-tcp-port, parent-connection-server-ip, parent-connection-server-tcp-port}. This 4-tuple corresponds to the equivalent 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port} in the twamp-server-ctrl-connection object. This 4-tuple allows the user to trace back from the TWAMP-Test session to the (parent) TWAMP-Control connection that negotiated this test session.

All data under twamp-reflector-test-session is read-only:

sid   An auto-allocated identifier for this TWAMP-Test session,
        that is unique within the context of this Server/Session-
        Reflector device only. This value will be communicated to
        the Control-Client that requested the test session in the SID
        field of the Accept-Session message.

sender-ip
        The IP address on the remote device, which is the source IP
        address used in the TWAMP-Test (UDP) packets belonging to
        this test session.

sender-udp-port
        The source UDP port used in the TWAMP-Test packets belonging
        to this test session.

reflector-ip
        The IP address of the local Session-Reflector device, which
        is the destination IP address used in the TWAMP-Test (UDP)
        packets belonging to this test session.

reflector-udp-port
        The destination UDP port number used in the TWAMP-Test (UDP)
        test packets belonging to this test session.

parent-connection-client-ip
        The IP address on the Control-Client device, which is the
        source IP address used in the TWAMP-Control (TCP) packets
        belonging to the parent control connection that negotiated
        this test session.

parent-connection-client-tcp-port
        The source TCP port number used in the TWAMP TCP control
        packets belonging to the parent control connection that
        negotiated this test session.

parent-connection-server-ip
        The IP address of the Server device, which is the destination
        IP address used in the TWAMP-Control (TCP) packets belonging
        to the parent control connection that negotiated this test
        session.

parent-connection-server-tcp-port
        The destination TCP port number used in the TWAMP-Control
        (TCP) packets belonging to the parent control connection that
        negotiated this test session.

dscp   The DSCP value present in the IP header of TWAMP-Test (UDP)
        packets belonging to this test session.

sent-packets
        The number of TWAMP-Test (UDP) response packets that have
        been sent by the Session-Reflector for this test session.

rcv-packets
        The number of TWAMP-Test (UDP) packets that have been
        received by the Session-Reflector for this test session.
        Since the Session-Reflector should respond to every test
        packet it receives, the sent-packets and rcv-packets values
        should always be identical.

last-sent-seq
        The value in the sequence number field of the last TWAMP-Test
        (UDP) response packet transmitted for this test session.

last-rcv-seq
        The value in the sequence number field of the last TWAMP-Test
        (UDP) packet received for this test session.

5. Data Model

   This section formally specifies the TWAMP data model using YANG.

5.1. YANG Tree Diagram

   This section presents a simplified graphical representation of the
   TWAMP data model using a YANG tree diagram. Readers should keep in
   mind that the limit of 72 characters per line forces us to introduce
   artificial line breaks in some tree diagram nodes.

```
   module: ietf-twamp
      +--rw twamp
         +--rw twamp-client! {control-client}?
         |  +--rw client-admin-state             boolean
         |  +--rw mode-preference-chain* [priority]
         |  |  +--rw priority uint16
         |  |  +--rw mode? mode
         |  +--rw key-chain* [key-id]
         |  |  +--rw key-id        string
         |  |  +--rw secret-key?   string
         |  +--rw twamp-client-ctrl-connection* [ctrl-connection-name]
         |     +--rw ctrl-connection-name     string
         |     +--rw client-ip?               inet:ip-address
         |     +--rw server-ip                inet:ip-address
         |     +--rw server-tcp-port?         inet:port-number
         |     +--rw dscp?                    inet:dscp
         |     +--rw key-id?                  string
         |     +--rw max-count?               uint32
         |     +--ro client-tcp-port?         inet:port-number
         |     +--ro server-start-time?       uint64
         |     +--ro ctrl-connection-state?   ctrl-connection-state
         |     +--ro selected-mode?           mode
         |     +--ro token?                   binary
         |     +--ro client-iv?               binary
         |     +--rw twamp-session-request* [test-session-name]
         |        +--rw test-session-name     string
         |        +--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 timeout?              uint64
         |        +--rw padding-length?       uint32
         |        +--rw dscp?                 inet:dscp
         |        +--rw start-time?           uint64
         |        +--rw repeat?               uint32
         |        +--rw repeat-interval?      uint32
         |        +--rw pm-reg-list* [pm-index]
```

```
|         | +--rw pm-index     uint16
|         +--ro test-session-state?   test-session-state
|         +--ro sid?                  string
+--rw twamp-server! {server}?
|  +--rw server-admin-state             boolean
|  +--rw server-tcp-port?               inet:port-number
|  +--rw servwait?                      uint32
|  +--rw dscp?                          inet:dscp
|  +--rw count?                         uint32
|  +--rw max-count?                     uint32
|  +--rw modes?                         mode
|  +--rw key-chain* [key-id]
|  |  +--rw key-id       string
|  |  +--rw secret-key?   string
|  +--ro twamp-server-ctrl-connection* \
|          [client-ip client-tcp-port \
|           server-ip server-tcp-port]
|     +--ro client-ip                   inet:ip-address
|     +--ro client-tcp-port             inet:port-number
|     +--ro server-ip                   inet:ip-address
|     +--ro server-tcp-port             inet:port-number
|     +--ro server-ctrl-connection-state? \
|             server-ctrl-connection-state
|     +--ro dscp?                       inet:dscp
|     +--ro selected-mode?              mode
|     +--ro key-id?                     string
|     +--ro count?                      uint32
|     +--ro max-count?                  uint32
|     +--ro salt?                       binary
|     +--ro server-iv?                  binary
|     +--ro challenge?                  binary
+--rw twamp-session-sender {session-sender}?
|  +--rw session-sender-admin-state        boolean
|  +--rw twamp-sender-test-session* [test-session-name]
|     +--rw test-session-name         string
|     +--ro ctrl-connection-name?     string
|     +--rw fill-mode?                fill-mode
|     +--rw number-of-packets?        uint32
|     +--rw (packet-distribution)?
|     |  +--:(periodic)
|     |  |  +--rw periodic-interval?        uint32
|     |  |  +--rw periodic-interval-units?   units
|     |  +--:(poisson)
|     |     +--rw lambda?                   uint32
|     |     +--rw lambda-units?             uint32
|     |     +--rw max-interval?             uint32
|     |     +--rw truncation-point-units?    units
|     +--ro sender-session-state?       sender-session-state
```

```
       |        +--ro sent-packets?               uint32
       |        +--ro rcv-packets?                uint32
       |        +--ro last-sent-seq?              uint32
       |        +--ro last-rcv-seq?               uint32
       +--rw twamp-session-reflector {session-reflector}?
          +--rw session-reflector-admin-state       boolean
          +--rw refwait?                         uint32
          +--ro twamp-reflector-test-session* \
                [sender-ip sender-udp-port \
                 reflector-ip reflector-udp-port]
             +--ro sid?                                string
             +--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 parent-connection-client-ip?        inet:ip-address
             +--ro parent-connection-client-tcp-port?  inet:port-number
             +--ro parent-connection-server-ip?        inet:ip-address
             +--ro parent-connection-server-tcp-port?  inet:port-number
             +--ro dscp?                               inet:dscp
             +--ro sent-packets?                       uint32
             +--ro rcv-packets?                        uint32
             +--ro last-sent-seq?                      uint32
             +--ro last-rcv-seq?                       uint32
```

5.2. YANG Module

This section presents the YANG module for the TWAMP data model
defined in this document.

```
<CODE BEGINS> file "ietf-twamp@2015-10-19.yang"
module ietf-twamp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-twamp";
   //namespace need to be assigned by IANA
  prefix "ietf-twamp";

  import ietf-inet-types {
    prefix inet;
  }

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

  contact "draft-cmzrjp-ippm-twamp-yang@tools.ietf.org";

  description "TWAMP Data Model";

  revision "2015-10-19" {
    description "01 version. RFC5357, RFC5618, RFC5938 and RFC6038
```

```
     is covered. draft-ietf-ippm-metric-registry is also considered";

   reference "draft-cmzrjp-ippm-twamp-yang";
   }

   feature control-client {
     description "This feature relates to the device functions as
     the TWAMP Control-Client.";
   }

   feature server {
     description "This feature relates to the device functions as
     the TWAMP Server.";
   }

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

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

   typedef ctrl-connection-state {
     type enumeration {
       enum active {
         description "Control session is active.";
       }
       enum idle {
         description "Control session is idle.";
       }
     }
     description "Control connection state";
   }

   typedef mode {
     type bits {
       bit unauthenticated {
         position "0";
         description "Unauthenticated";
       }
       bit authenticated {
         position "1";
         description "Authenticated";
       }
       bit encrypted {
```

```
            position "2";
            description "Encrypted";
          }
          bit unauth-test-encrpyt-control {
            position "3";
            description "Mixed Security Mode per RFC 5618. Test
            protocol security mode in Unauthenticated mode,
            Control protocol in Encrypted mode.";
          }
          bit individual-session-control {
            position "4";
            description "Individual session control per RFC5938.";
          }
          bit reflect-octets {
            position "5";
            description "Reflect octets capability per RFC6038.";
          }
          bit symmetrical-size {
            position "6";
            description "Symmetrical size per RFC6038.";
          }
        }
        description "Authentication mode bit mask";
      }

      typedef test-session-state {
        type enumeration {
          enum ok {
            value 0;
            description "Test session is accepted.";
          }
          enum failed {
            value 1;
            description "Failure, reason unspecified (catch-all).";
          }
          enum internal-error {
            value 2;
            description "Internal error.";
          }
          enum not-supported {
            value 3;
            description "Some aspect of request is not supported.";
          }
          enum permanent-resource-limit {
            value 4;
            description "Cannot perform request due to
            permanent resource limitations.";
          }
```

```
      enum temp-resource-limit {
        value 5;
        description "Cannot perform request due to
        temporary resource limitations.";
      }
    }
    description "Test session state";
  }

  typedef server-ctrl-connection-state {
    type enumeration {
      enum "active" {
        description "Active";
      }
      enum "servwait" {
        description "Servwait";
      }
    }
    description "Server control connection state";
  }

  typedef fill-mode {
    type enumeration {
      enum zero {
        description "Zero";
      }
      enum random {
        description "Random";
      }
    }
    description "Indicates whether the padding added to the
    UDP test packets will contain pseudo-random numbers, or
    whether it should consist of all zeroes.";
  }

  typedef units  {
    type enumeration {
      enum seconds {
        description "Seconds";
      }
      enum milliseconds {
        description "Milliseconds";
      }
      enum microseconds {
        description "Microseconds";
      }
      enum nanoseconds {
        description "Nanoseconds";
```

```
          }
        }
        description "Time units";
      }

      typedef sender-session-state {
        type enumeration {
          enum setup {
            description "Test session is active.";
          }
          enum failure {
            description "Test session is idle.";
          }
        }
        description "Sender session state.";
      }

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

      container twamp {
        description "Top level container";
        container twamp-client {
          if-feature control-client;
          presence "twamp-client";
          description "Twamp client container";
          leaf client-admin-state {
```

```
            type boolean;
            mandatory "true";
            description "Indicates whether this device is allowed to run
            TWAMP to initiate control sessions";
          }

          list mode-preference-chain {
            key "priority";
            unique "mode";
            leaf priority {
              type uint16;
              description "priority";
            }
            leaf mode {
              type mode;
              description "Authentication mode bit mask";
            }
            description "Authentication mode preference";
          }

          list key-chain {
            key "key-id";
            leaf key-id {
              type string {
                length "1..80";
              }
              description "Key ID";
            }
            leaf secret-key {
              type string;
              description "Secret key";
            }
            description "Key chain";
          }

          list twamp-client-ctrl-connection {
            key "ctrl-connection-name";
            description "Twamp client control connections";
            leaf ctrl-connection-name {
              type string;
              description "A unique name used as a key to identify this
                 individual TWAMP control connection on the
                 Control-Client device.";
            }
            leaf client-ip {
              type inet:ip-address;
              description "Client IP address";
            }
```

```
            leaf server-ip {
              type inet:ip-address;
              mandatory "true";
              description "Server IP address";
            }
            leaf server-tcp-port {
              type inet:port-number;
              default "862";
              description "Server tcp port";
            }
            leaf dscp{
              type inet:dscp;
              default "0";
              description "The DSCP value to be placed in the IP header
                of the TWAMP TCP Control packets generated
                by the Control-Client";
            }
            leaf key-id {
              type string {
                length "1..80";
              }
              description "Key ID";
            }
            leaf max-count {
              type uint32 {
                range 1024..4294967295;
              }
              default 32768;
              description "Max count value.";
            }
            leaf client-tcp-port {
              type inet:port-number;
              config "false";
              description "Client TCP port";
            }
            leaf server-start-time {
              type uint64;
              config "false";
              description "The Start-Time advertized by the Server in
              the Server-Start message";
            }
            leaf ctrl-connection-state {
              type ctrl-connection-state;
              config "false";
              description "Control connection state";
            }
            leaf selected-mode {
              type mode;
```

```
                config "false";
                description "The TWAMP mode that the Control-Client has
                chosen for this control connection as set in the Mode
                field of the Set-Up-Response message";
            }
            leaf token {
                type binary {
                  length "64";
                }
                config "false";
                description "64 octets, containing the concatenation of a
                   16-octet challenge, a 16-octet AES Session-key used
                   for encryption, and a 32-octet HMAC-SHA1 Session-key
                   used for authentication";
            }
            leaf client-iv{
              type binary {
                length "16";
              }
              config "false";
              description "16 octets, Client-IV is generated randomly
                by the Control-Client.";
            }

            list twamp-session-request {
              key "test-session-name";
              description "Twamp session requests";
              leaf test-session-name {
                type string;
                description "A unique name for this test session to be
                used as a key for this test session on the
                Control-Client.";
              }
              leaf sender-ip {
                type inet:ip-address;
                description "Sender IP address";
              }
              leaf sender-udp-port {
                type inet:port-number;
                description "Sender UDP port";
              }
              leaf reflector-ip {
                type inet:ip-address;
                mandatory "true";
                description "Reflector IP address.";
              }
              leaf reflector-udp-port {
                type inet:port-number;
```

```
                    description "Reflector UDP port. If this value is not
                    set, the device shall use the same port number as
                    defined in the server-tcp-port parameter of this
                    twamp-session-request's
                    parent client-control-connection.";
                  }
                  leaf timeout {
                    type uint64;
                    default "2";
                    description "The time (in seconds)Session-Reflector MUST
                    wait after receiving a Stop-Session message.";
                  }
                  leaf padding-length {
                    type uint32{
                      range "64..4096";
                    }
                    description "The number of bytes of padding that should
                         be added to the UDP test packets generated by the
                         sender. Jumbo sized packets supported.";
                  }
                  leaf dscp {
                    type inet:dscp;
                    description "The DSCP value to be placed in the UDP
                    header of TWAMP-Test packets generated by the
                    Session-Sender, and in the UDP header of the TWAMP-Test
                    response packets generated by the Session-Reflector
                    for this test session.";
                  }
                  leaf start-time {
                    type uint64;
                    default "0";
                    description "Time when the session is to be started
                    (but not before the Start-Sessions command is issued).
                    This value is placed in the Start Time field of the
                    Request-TW-Session message. The default value of 0
                    indicates that the session will be started as soon
                    as the Start-Sessions message is received.";
                  }
                  leaf repeat {
                    type uint32;
                    default "0";
                    description "Determines if the test session is to be
                    run repeatedly. The default value of repeat is 0,
                    indicating that once the session has completed, it
                    will not be renegotiated and restarted";
                  }
                  leaf repeat-interval  {
                    when "../repeat!='0'" {
```

```
              description "When repeat is not 0, the test is to be
                repeated";
            }
            type uint32;
            description "Repeat interval (in minutes)";
          }

          list pm-reg-list {
            key "pm-index";
            leaf pm-index {
              type uint16;
              description "One or more Numerical index values of a
                Registered Metric in the Performance Metric Registry";
            }
            description "A list of one or more pm-index values,
            which communicate packet stream characteristics and one
            or more metrics to be measured.";
          }
          leaf test-session-state {
            type test-session-state;
            config "false";
            description "Test session state";
          }
          leaf sid{
            type string;
            config "false";
            description "The SID allocated by the Server for
            this test session";
          }
        }
      }
    }
  }

  container twamp-server{
    if-feature server;
    presence "twamp-server";
    description "Twamp sever container";
    leaf server-admin-state{
      type boolean;
      mandatory "true";
      description "Indicates whether this device is allowed to run
        TWAMP to respond to control sessions";
    }
    leaf server-tcp-port {
      type inet:port-number;
      default "862";
      description "This parameter defines the well known TCP port
      number that is used by TWAMP.";
```

```
        }
        leaf servwait {
          type uint32 {
            range 1..604800;
          }
          default 900;
          description "SERVWAIT (TWAMP Control (TCP) session timeout),
            default value is 900";
        }
        leaf dscp {
          type inet:dscp;
          description "The DSCP value to be placed in the IP header of
          TCP TWAMP-Control packets generated by the Server";
        }
        leaf count {
          type uint32 {
            range 1024..4294967295;
          }
          description "Parameter used in deriving a key from a
          shared secret ";
        }
        leaf max-count {
          type uint32 {
            range 1024..4294967295;
          }
          default 32768;
          description "Max count value.";
        }
        leaf modes {
          type mode;
          description "The bit mask of TWAMP Modes this Server
          instance is willing to support.";
        }

        list key-chain {
          key "key-id";
          leaf key-id {
            type string {
              length "1..80";
            }
            description "Key IDs.";
          }
          leaf secret-key {
            type string;
            description "Secret keys.";
          }
          description "KeyIDs with the respective secret keys.";
        }
```

```
list twamp-server-ctrl-connection {
  key "client-ip client-tcp-port server-ip server-tcp-port";
  config "false";
  description "Twamp server control connections";
  leaf client-ip {
    type inet:ip-address;
    description "Client IP address";
  }
  leaf client-tcp-port {
    type inet:port-number;
    description "Client TCP port";
  }
  leaf server-ip {
    type inet:ip-address;
    description "Server IP address";
  }
  leaf server-tcp-port {
    type inet:port-number;
    description "Server TCP port";
  }
  leaf server-ctrl-connection-state {
    type server-ctrl-connection-state;
    description "Server control connection state";
  }
  leaf dscp {
    type inet:dscp;
    description "The DSCP value used in the IP header of the
    TCP control packets sent by the Server for this control
    connection. This will usually be the same value as is
    configured for twamp-server:dscp under the twamp-server.
    However, in the event that the user re-configures
    twamp-server:dscp after this control connection is already
    in progress, this read-only value will show the actual
    dscp value in use by this control connection.";
  }
  leaf selected-mode {
    type mode;
    description "The mode that was chosen for this control
    connection as set in the Mode field of the
    Set-Up-Response message.";
  }
  leaf key-id {
    type string {
      length "1..80";
    }
    description "The key-id value that is in use by this
            control connection.";
  }
```

```
            leaf count {
              type uint32 {
                range 1024..4294967295;
              }
              description "The count value that is in use by this control
              connection. This will usually be the same value as is
              configured under twamp-server. However, in the event that
              the user re-configured twamp-server:count after this
              control connection is already in progress, this read-only
              value will show the different count that is in use for
              this control connection.";
            }
            leaf max-count {
              type uint32 {
                range 1024..4294967295;
              }
              description "The max-count value that is in use by this
              control connection. This will usually be the same value
              as is configured under twamp-server. However, in the
              event that the user re-configured twamp-server:max-count
              after this control connection is already in progress,
              this read-only value will show the different max-count
              that is in use for this control connection.";
            }
            leaf salt{
              type binary {
                length "16";
              }
              description "Salt MUST be generated pseudo-randomly";
            }
            leaf server-iv {
              type binary {
                length "16";
              }
              description "16 octets, Server-IV is generated randomly
              by the Control-Client.";
            }
            leaf challenge {
              type binary {
                length "16";
              }
              description "Challenge is a random sequence of octets
              generated by the Server";
            }
          }
        }

      container twamp-session-sender{
```

```
          if-feature session-sender;
          description "Twamp session sender container";
          leaf session-sender-admin-state {
            type boolean;
            mandatory "true";
            description "Indicates whether this device is allowed to run
            TWAMP to initiate test sessions";
          }
          list twamp-sender-test-session{
            key "test-session-name";
            description "Twamp sender test sessions";
            leaf test-session-name {
              type string;
              description "A unique name for this test session to be
              used as a key for this test session by the Session-Sender
              logical entity.";
            }
            leaf ctrl-connection-name {
              type string;
              config "false";
              description "The name of the parent control connection
              that is responsible for negotiating this test session.";
            }
            leaf fill-mode {
              type fill-mode;
              default zero;
              description "Indicates whether the padding added to the
              UDP test packets will contain pseudo-random numbers, or
              whether it should consist of all zeroes.";
            }
            leaf number-of-packets {
              type uint32;
              description "The overall number of UDP test packets to be
                transmitted by the sender for this test session.";
            }
            choice packet-distribution {
              description "Packet distributions, poisson or periodic";
              case periodic {
                leaf periodic-interval {
                  type uint32;
                  description "Periodic interval";
                }
                leaf periodic-interval-units  {
                  type units;
                  description "Periodic interval units";
                }
              }
              case poisson {
```

```
            leaf lambda{
              type uint32;
              description "The average rate of
              packet transmission.";
            }
            leaf lambda-units{
              type uint32;
              description "Lambda units.";
            }
            leaf max-interval{
              type uint32;
              description "maximum time between packet
              transmissions.";
            }
            leaf truncation-point-units{
              type units;
              description "Truncation point units";
            }
          }
        }
        leaf sender-session-state {
          type sender-session-state;
          config "false";
          description "Sender session state.";
        }
        uses maintenance-statistics;
      }
    }

    container twamp-session-reflector {
      if-feature session-reflector;
      description "Twamp session reflector container";
      leaf session-reflector-admin-state {
        type boolean;
        mandatory "true";
        description "Indicates whether this device is allowed to run
        TWAMP to respond to test sessions";
      }
      leaf refwait {
        type uint32 {
          range 1..604800;
        }
        default 900;
        description "REFWAIT (TWAMP test session timeout),
          the default value is 900";
      }

      list twamp-reflector-test-session {
```

```
            key "sender-ip sender-udp-port reflector-ip
                reflector-udp-port";
            config "false";
            description "Twamp reflector test sessions";
            leaf sid{
              type string;
              description "An auto-allocated identifier for this test
              session, that is unique within the context of this
              Server/Session-Reflector device only. ";
            }
            leaf sender-ip {
              type inet:ip-address;
              description "Sender IP address.";
            }
            leaf sender-udp-port {
              type inet:port-number;
              description "Sender UDP port.";
            }
            leaf reflector-ip {
              type inet:ip-address;
              description "Reflector IP address.";
            }
            leaf reflector-udp-port {
              type inet:port-number;
              description "Reflector UDP port.";
            }
            leaf parent-connection-client-ip {
              type inet:ip-address;
              description "Parent connction client IP address.";
            }
            leaf parent-connection-client-tcp-port {
              type inet:port-number;
              description "Parent connection client TCP port.";
            }
            leaf parent-connection-server-ip {
              type inet:ip-address;
              description "Parent connection server IP address.";
            }
            leaf parent-connection-server-tcp-port {
              type inet:port-number;
              description "Parent connection server TCP port";
            }
            leaf dscp {
              type inet:dscp;
              description "The DSCP value present in the IP header of
              TWAMP UDP test packets belonging to this test session.";
            }
            uses maintenance-statistics;
```

```
            }
          }
        }
      }
```

    <CODE ENDS>

6. Data Model Examples

   This section presents a simple but complete example of configuring
   all four entities in Figure 1, based on the YANG module specified in
   Section 5. The example is illustrative in nature, but aims to be
   self-contained, i.e. were it to be executed in a real TWAMP
   implementation it would lead to a correctly configured test session.
   A more elaborated example, which also includes authentication
   parameters, is provided in Appendix A.

6.1. Control-Client

   The following configuration example shows a Control-Client with
   client-admin-state enabled. In a real implementation following
   Figure 2 this would permit the initiation of TWAMP-Control
   connections and TWAMP-Test sessions.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-client>
      <client-admin-state>True</client-admin-state>
   </twamp-client>
</twamp>
```

   The following configuration example shows a Control-Client with two
   instances of twamp-client-ctrl-connection, one called "RouterA" and
   another called "RouterB". Each TWAMP-Control connection is to a
   different Server. The control connection named "RouterA" has two
   test session requests. The TWAMP-Control connection named "RouterB"
   has no TWAMP-Test session requests.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-client>

       <twamp-client-ctrl-connection>
           <ctrl-connection-name>RouterA</ctrl-connection-name>
           <client-ip>203.0.113.1</client-ip>
           <server-ip>203.0.113.2</server-ip>
           <twamp-session-request>
               <test-session-name>Test1</test-session-name>
               <sender-ip>10.1.1.1</sender-ip>
               <sender-udp-port>4000</sender-udp-port>
               <reflector-ip>10.1.1.2</reflector-ip>
               <reflector-udp-port>5000</reflector-udp-port>
               <start-time>0</start-time>
           </twamp-session-request>
           <twamp-session-request>
               <test-session-name>Test2</test-session-name>
               <sender-ip>203.0.113.1</sender-ip>
               <sender-udp-port>4001</sender-udp-port>
               <reflector-ip>203.0.113.2</reflector-ip>
               <reflector-udp-port>5001</reflector-udp-port>
               <start-time>0</start-time>
           </twamp-session-request>
       </twamp-client-ctrl-connection>

       <twamp-client-ctrl-connection>
           <ctrl-connection-name>RouterB</ctrl-connection-name>
           <client-ip>203.0.113.1</client-ip>
           <server-ip>203.0.113.3</server-ip>
       </twamp-client-ctrl-connection>

   </twamp-client>
</twamp>
```

6.2. Server

This configuration example shows a Server with server-admin-state
enabled, which permits a device following Figure 2 to respond to
TWAMP-Control connections and TWAMP-Test sessions.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-server>
      <server-admin-state>True</server-admin-state>
   </twamp-server>
</twamp>
```

The following example presents a Server with the TWAMP-Control
connection corresponding to the control connection name (ctrl-
connection-name) "RouterA" presented in Section 6.1.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-server>

      <twamp-server-ctrl-connection>
         <client-ip>203.0.113.1</client-ip>
         <client-tcp-port>16341</client-tcp-port>
         <server-ip>203.0.113.2</server-ip>
         <server-tcp-port>862</server-tcp-port>
         <server-ctrl-connection-state>
            active
         </server-ctrl-connection-state>
      </twamp-server-ctrl-connection>

   </twamp-server>
</twamp>
```

6.3. Session-Sender

The following configuration example shows a Session-Sender with the
two TWAMP-Test sessions presented in Section 6.1.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-session-sender>

      <twamp-sender-test-session>
         <test-session-name>Test1</test-session-name>// read-only
         <ctrl-connection-name>RouterA</ctrl-connection-name>
         <number-of-packets>900</number-of-packets>
         <packet-distribution>
            <periodic-interval>1</periodic-interval>
            <periodic-interval-units>seconds</periodic-interval-units>
         </packet-distribution>
      </twamp-sender-test-session>

      <twamp-sender-test-session>
         <test-session-name>Test2</test-session-name>
         <ctrl-connection-name>
           RouterA
         </ctrl-connection-name> // read-only
         <number-of-packets>900</number-of-packets>
         <packet-distribution>
            <lambda>1</lambda>
            <lambda-units>1</lambda-units>
            <max-interval>2</max-interval>
            <truncation-point-units>seconds</truncation-point-units>
         </packet-distribution>
      </twamp-sender-test-session>

   </twamp-session-sender>
 </twamp>
```

6.4. Session-Reflector

The following example shows the two Session-Reflector TWAMP-Test
sessions corresponding to the test sessions presented in Section 6.3.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-session-reflector>

      <twamp-reflector-test-session>
        <sid>1232</sid>
        <sender-ip>10.1.1.1</sender-ip>
        <reflector-ip>10.1.1.2</reflector-ip>
        <sender-udp-port>4000</sender-udp-port>
        <reflector-udp-port>5000</reflector-udp-port>
        <parent-connection-client-ip>
           203.0.113.1
         </parent-connection-client-ip>
        <parent-connection-client-tcp-port>
```

```
               16341
            </parent-connection-client-tcp-port>
          <parent-connection-server-ip>
               203.0.113.2
            </parent-connection-server-ip>
          <parent-connection-server-tcp-port>
               862
            </parent-connection-server-tcp-port>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </twamp-reflector-test-session>

        <twamp-reflector-test-session>
          <sid>178943</sid>
          <sender-ip>203.0.113.1</sender-ip>
          <reflector-ip>192.68.0.2</reflector-ip>
          <sender-udp-port>4001</sender-udp-port>
          <parent-connection-client-ip>
               203.0.113.1
            </parent-connection-client-ip>
          <parent-connection-client-tcp-port>
               16341
            </parent-connection-client-tcp-port>
          <parent-connection-server-ip>
               203.0.113.2
            </parent-connection-server-ip>
          <parent-connection-server-tcp-port>
             862
          </parent-connection-server-tcp-port>
          <reflector-udp-port>5001</reflector-udp-port>
          <sent-packets>21</sent-packets>
          <rcv-packets>21</rcv-packets>
          <last-sent-seq>20</last-sent-seq>
          <last-rcv-seq>20</last-rcv-seq>
        </twamp-reflector-test-session>

      </twamp-session-reflector>
   </twamp>
```

7. Security Considerations

   TBD

8. 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-twamp

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-twamp

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

prefix: twamp

reference: RFC XXXX

9. Acknowledgements

We thank Gregory Mirsky, Kevin D'Souza, and Robert Sherman for their
thorough and constructive reviews, comments and text suggestions.

Haoxing Shen contributed to the definition of the YANG module in
Section 5.

10. References

10.1. 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>.

   [RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network
              performance measurement with periodic streams", RFC 3432,
              DOI 10.17487/RFC3432, November 2002,
              <http://www.rfc-editor.org/info/rfc3432>.

   [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
              DOI 10.17487/RFC3688, January 2004,
              <http://www.rfc-editor.org/info/rfc3688>.

   [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M.
              Zekauskas, "A One-way Active Measurement Protocol
              (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006,
              <http://www.rfc-editor.org/info/rfc4656>.

   [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J.
              Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)",
              RFC 5357, DOI 10.17487/RFC5357, October 2008,
              <http://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,
              <http://www.rfc-editor.org/info/rfc6020>.

   [RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement
              Protocol (TWAMP) Reflect Octets and Symmetrical Size
              Features", RFC 6038, DOI 10.17487/RFC6038, October 2010,
              <http://www.rfc-editor.org/info/rfc6038>.

10.2. Informative References

   [I-D.ietf-ippm-metric-registry]
              Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A.
              Akhter, "Registry for Performance Metrics", draft-ietf-
              ippm-metric-registry-05 (work in progress), October 2015.

   [I-D.ietf-netconf-restconf]
              Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", draft-ietf-netconf-restconf-08 (work in
              progress), October 2015.

   [I-D.unify-nfvrg-challenges]
              Szabo, R., Csaszar, A., Pentikousis, K., Kind, M., Daino,
              D., Qiang, Z., and H. Woesner, "Unifying Carrier and Cloud
              Networks: Problem Statement and Challenges", draft-unify-
              nfvrg-challenges-02 (work in progress), July 2015.

[I-D.unify-nfvrg-devops]
          Meirosu, C., Manzalini, A., Steinert, R., Marchetto, G.,
          Papafili, I., Pentikousis, K., and S. Wright, "DevOps for
          Software-Defined Telecom Infrastructures", draft-unify-
          nfvrg-devops-03 (work in progress), October 2015.

[NSC]   John, W., Pentikousis, K., et al., "Research directions in
          network service chaining", Proc. SDN for Future Networks
          and Services (SDN4FNS), Trento, Italy IEEE, November 2013.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography
          Specification Version 2.0", RFC 2898,
          DOI 10.17487/RFC2898, September 2000,
          <http://www.rfc-editor.org/info/rfc2898>.

[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>.

[RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the
          Two-Way Active Measurement Protocol (TWAMP)", RFC 5618,
          DOI 10.17487/RFC5618, August 2009,
          <http://www.rfc-editor.org/info/rfc5618>.

[RFC5938] Morton, A. and M. Chiba, "Individual Session Control
          Feature for the Two-Way Active Measurement Protocol
          (TWAMP)", RFC 5938, DOI 10.17487/RFC5938, August 2010,
          <http://www.rfc-editor.org/info/rfc5938>.

[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,
          <http://www.rfc-editor.org/info/rfc6241>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S.,
          Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-
          Defined Networking (SDN): Layers and Architecture
          Terminology", RFC 7426, DOI 10.17487/RFC7426, January
          2015, <http://www.rfc-editor.org/info/rfc7426>.

Appendix A. Detailed Data Model Examples

This appendix extends the example presented in Section 6 by
configuring more fields such as authentication parameters, dscp
values and so on.

A.1. Control-Client

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-client>

      <client-admin-state>True</client-admin-state>

      <mode-preference-chain>
         <priority>0</priority>
         <mode>0x00000002</mode>
      </mode-preference-chain>
      <mode-preference-chain>
         <priority>1</priority>
         <mode>0x00000001</mode>
      </mode-preference-chain>

      <keychain>
         <keyid>KeyClient1ToRouterA</keyid>
         <secret-key>secret1</secret-key>
      </keychain>
      <keychain>
         <keyid>KeyForRouterB</keyid>
         <secret-key>secret2</secret-key>
      </keychain>

      <twamp-client-ctrl-connection>
         <ctrl-connection-name>RouterA</ctrl-connection-name>
         <client-ip>203.0.113.1</client-ip>
         <server-ip>203.0.113.2</server-ip>
         <dscp>32</dscp>
         <key-id>KeyClient1ToRouterA</key-id>
         <twamp-session-request>
            <test-session-name>Test1</test-session-name>
            <sender-ip>10.1.1.1</sender-ip>
            <sender-udp-port>4000</sender-udp-port>
            <reflector-ip>10.1.1.2</reflector-ip>
            <reflector-udp-port>5000</reflector-udp-port>
            <padding-length>0</padding-length>
            <start-time>0</start-time>
            <test-session-state>ok</test-session-state>
            <sid>1232</sid>
         </twamp-session-request>
         <twamp-session-request>
            <test-session-name>Test2</test-session-name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>4001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>5001</reflector-udp-port>
```

```
                <paddingLenth>32</paddingLenth>
                <start-time>0</start-time>
                <test-session-state>ok</test-session-state>
                <sid>178943</sid>
            </twamp-session-request>
        </twamp-client-ctrl-connection>

    </twamp-client>
  </twamp>
```

A.2. Server

```
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <twamp-server>

        <server-admin-state>True</server-admin-state>
        <servwait>1800</servwait>
        <dscp>32</dscp>
        <modes>0x00000003</modes>
        <count>256</count>

        <keychain>
            <keyid>KeyClient1ToRouterA</keyid>
            <secret-key>secret1</secret-key>
        </keychain>
        <keychain>
            <keyid>KeyClient10ToRouterA</keyid>
            <secret-key>secret10</secret-key>
        </keychain>

        <twamp-server-ctrl-connection>
            <client-ip>203.0.113.1</client-ip>
            <client-tcp-port>16341</client-tcp-port>
            <server-ip>203.0.113.2</server-ip>
            <server-tcp-port>862</server-tcp-port>
            <server-ctrl-connection-state>
               active
             </server-ctrl-connection-state>
            <dscp>32</dscp>
            <selected-mode>0x00000002</selected-mode>
            <key-id>KeyClient1ToRouterA</key-id>
            <count>1024</count>
        </twamp-server-ctrl-connection>

    </twamp-server>
  </twamp>
```

A.3. Session-Sender

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-session-sender>

      <twamp-sender-test-session>
         <test-session-name>Test1</test-session-name> // read-only
         <ctrl-connection-name>RouterA</ctrl-connection-name>
         <dscp>32</dscp>
         <fill-mode>zero</fill-mode>
         <number-of-packets>900</number-of-packets>
         <packet-distribution>
            <periodic-interval>1</periodic-interval>
            <periodic-interval-units>seconds</periodic-interval-units>
         </packet-distribution>
         <sender-session-state>Active</sender-session-state>
         <sent-packets>2</sent-packets>
         <rcv-packets>2</rcv-packets>
         <last-sent-seq>1</last-sent-seq>
         <last-rcv-seq>1</last-rcv-seq>
      </twamp-sender-test-session>

      <twamp-sender-test-session>
         <test-session-name>Test2</test-session-name>
         <ctrl-connection-name>
           RouterA
         </ctrl-connection-name> // read-only
         <dscp>32</dscp>
         <fill-mode>random</fill-mode>
         <number-of-packets>900</number-of-packets>
         <packet-distribution>
           <lambda>1</lambda>
           <lambda-units>1</lambda-units>
           <max-interval>2</max-interval>
           <truncation-point-units>seconds</truncation-point-units>
         </packet-distribution>
         <sender-session-state>Active</sender-session-state>
         <sent-packets>21</sent-packets>
         <rcv-packets>21</rcv-packets>
         <last-sent-seq>20</last-sent-seq>
         <last-rcv-seq>20</last-rcv-seq>
      </twamp-sender-test-session>

   </twamp-session-sender>
</twamp>
```

A.4. Session-Reflector

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-session-reflector>

      <twamp-reflector-test-session>
         <sid>1232</sid>
         <sender-ip>10.1.1.1</sender-ip>
         <reflector-ip>10.1.1.2</reflector-ip>
         <sender-udp-port>4000</sender-udp-port>
         <reflector-udp-port>5000</reflector-udp-port>
         <parent-connection-client-ip>
            203.0.113.1
         </parent-connection-client-ip>
         <parent-connection-client-tcp-port>
            16341
         </parent-connection-client-tcp-port>
         <parent-connection-server-ip>
            203.0.113.2
         </parent-connection-server-ip>
         <parent-connection-server-tcp-port>
            862
         </parent-connection-server-tcp-port>
         <dscp>32</dscp>
         <sent-packets>2</sent-packets>
         <rcv-packets>2</rcv-packets>
         <last-sent-seq>1</last-sent-seq>
         <last-rcv-seq>1</last-rcv-seq>
      </twamp-reflector-test-session>

      <twamp-reflector-test-session>
         <sid>178943</sid>
         <sender-ip>203.0.113.1</sender-ip>
         <reflector-ip>192.68.0.2</reflector-ip>
         <sender-udp-port>4001</sender-udp-port>
         <parent-connection-client-ip>
            203.0.113.1
         </parent-connection-client-ip>
         <parent-connection-client-tcp-port>
            16341
         </parent-connection-client-tcp-port>
         <parent-connection-server-ip>
            203.0.113.2
         </parent-connection-server-ip>
         <parent-connection-server-tcp-port>
            862
         </parent-connection-server-tcp-port>
         <reflector-udp-port>5001</reflector-udp-port>
```

```
               <dscp>32</dscp>
               <sent-packets>21</sent-packets>
               <rcv-packets>21</rcv-packets>
               <last-sent-seq>20</last-sent-seq>
               <last-rcv-seq>20</last-rcv-seq>
            </twamp-reflector-test-session>

      </twamp-session-reflector>
   </twamp>
```

Appendix B. TWAMP Operational Commands

This document is targeted at configuration details for TWAMP.
Operational actions such as how TWAMP sessions are started/stopped,
how results are retrieved, or stored results are cleared, and so on,
are not addressed by this configuration model and are out of scope of
this document.

TWAMP operational commands could be performed programmatically or
manually, e.g. using a command-line interface (CLI). With respect to
programmability, YANG can be used to define NETCONF Remote Procedure
Calls (RPC), therefore it would be possible to define RPC operations
for actions such as starting or stopping control or test sessions or
groups of sessions; retrieving results; clearing stored results, and
so on.

However, [RFC5357] does not attempt to describe such operational
actions, and it is likely that different TWAMP implementations could
support different sets of operational commands, with different
restrictions. Therefore, this document considers it the
responsibility of the individual implementation to define its
corresponding TWAMP operational commands data model.

Authors' Addresses

   Ruth Civil
   Ciena Corporation
   307 Legget Drive
   Kanata, ON K2K 3C8
   Canada

   Email: gcivil@ciena.com
   URI: www.ciena.com

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

    Phone: +1 732 420 1571
    Fax: +1 732 368 1192
    Email: acmorton@att.com
    URI: http://home.comcast.net/~acmacm/

    Lianshu Zheng
    Huawei Technologies
    China

    Email: vero.zheng@huawei.com

    Reshad Rahman
    Cisco Systems
    2000 Innovation Drive
    Kanata, ON K2K 3E8
    Canada

    Email: rrahman@cisco.com

    Mahesh Jethanandani
    Ciena Corporation
    3939 North 1st Street
    San Jose, CA 95134
    USA

    Email: mjethanandani@gmail.com
    URI: www.ciena.com

    Kostas Pentikousis (editor)
    EICT GmbH
    EUREF-Campus Haus 13
    Torgauer Strasse 12-15
    10829 Berlin
    Germany

    Email: k.pentikousis@eict.de