```
module ietf-dots-telemetry {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-telemetry";
  prefix dots-telemetry;

  import ietf-dots-signal-channel {
    prefix ietf-signal;
    reference
      "RFC 8782: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Signal Channel Specification";
  }
  import ietf-dots-data-channel {
    prefix ietf-data;
    reference
      "RFC 8783: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Data Channel Specification";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "Section 6.2 of RFC 8345: A YANG Data Model for Network
       Topologies";
  }
  import ietf-yang-structure-ext {
    prefix sx;
    reference
      "RFC 8791: YANG Data Structure Extensions";
  }

  organization
    "IETF DDoS Open Threat Signaling (DOTS) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/dots/>
     WG List:  <mailto:dots@ietf.org>

     Author:  Mohamed Boucadair
              <mailto:mohamed.boucadair@orange.com>

     Author:  Konda, Tirumaleswar Reddy
              <mailto:TirumaleswarReddy_Konda@McAfee.com>";
  description
    "This module contains YANG definitions for the signaling
     of DOTS telemetry exchanged between a DOTS client and
     a DOTS server, by means of the DOTS signal channel.

     Copyright (c) 2020 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
```

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

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

  revision 2020-05-04 2020-07-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Telemetry";
  }

  feature dots-telemetry {
    description
      "This feature indicates that the DOTS signal channel is able
       to convey DOTS telemetry data between DOTS clients and
       servers.";
  }

  typedef attack-severity {
    type enumeration {
      enum none {
        value 1;
        description
          "No effect on the DOTS client domain.";
      }
      enum low {
        value 2;
        description
          "Minimal effect on the DOTS client domain.";
      }
      enum medium {
        value 3;
        description
          "A subset of DOTS client domain resources are
           out of service.";
      }
      enum high {
        value 4;
        description
          "The DOTS client domain is under extremly severe
           conditions.";
      }
      enum unknown {
        value 5;
        description
          "The impact of the attack is not known.";
      }
    }
    description
```

```
      "Enumeration for attack severity.";
    reference
      "RFC 7970: The Incident Object Description Exchange
                 Format Version 2";
  }

  typedef unit-type {
    type enumeration {
      enum packet-ps {
        value 1;
        description
          "Packets per second (pps).";
      }
      enum bit-ps {
        value 2;
        description
          "Bits per Second (bit/s).";
      }
      enum byte-ps {
        value 3;
        description
          "Bytes per second (Byte/s).";
      }
    }
    description
      "Enumeration to indicate which unit type is used.";
  }

  typedef unit {
    type enumeration {
      enum packet-ps {
        value 1;
        description
          "Packets per second (pps).";
      }
      enum bit-ps {
        value 2;
        description
          "Bits per Second (bps).";
      }
      enum byte-ps {
        value 3;
        description
          "Bytes per second (Bps).";
      }
      enum kilopacket-ps {
        value 4;
        description
          "Kilo packets per second (kpps).";
      }
      enum kilobit-ps {
        value 5;
        description
          "Kilobits per second (kbps).";
      }
      enum kilobyte-ps {
        value 6;
```

```
        description
          "Kilobytes per second (kBps).";
      }
      enum megapacket-ps {
        value 7;
        description
          "Mega packets per second (Mpps).";
      }
      enum megabit-ps {
        value 8;
        description
          "Megabits per second (Mbps).";
      }
      enum megabyte-ps {
        value 9;
        description
          "Megabytes per second (MBps).";
      }
      enum gigapacket-ps {
        value 10;
        description
          "Giga packets per second (Gpps).";
      }
      enum gigabit-ps {
        value 11;
        description
          "Gigabits per second (Gbps).";
      }
      enum gigabyte-ps {
        value 12;
        description
          "Gigabytes per second (GBps).";
      }
      enum terapacket-ps {
        value 13;
        description
          "Tera packets per second (Tpps).";
      }
      enum terabit-ps {
        value 14;
        description
          "Terabits per second (Tbps).";
      }
      enum terabyte-ps {
        value 15;
        description
          "Terabytes per second (TBps).";
      }
    }
    description
      "Enumeration to indicate which unit is used.";
  }

  typedef interval {
    type enumeration {
      enum hour {
        value 1;
```

```
        description
          "Hour.";
      }
      enum day {
        value 2;
        description
          "Day.";
      }
      enum week {
        value 3;
        description
          "Week.";
      }
      enum month {
        value 4;
        description
          "Month.";
      }
    }
    description
      "Enumeration to indicate the overall measurement period.";
  }

  typedef sample {
    type enumeration {
      enum second {
        value 1;
        description
          " A one second measurement period.";
      }
      enum 5-seconds {
        value 2;
        description
          "5 seconds measurement period.";
      }
      enum 30-seconds {
        value 3;
        description
          "30 seconds measurement period.";
      }
      enum minute {
        value 4;
        description
          "One minute measurement period.";
      }
      enum 5-minutes {
        value 5;
        description
          "5 minutes measurement period.";
      }
      enum 10-minutes {
        value 6;
        description
          "10 minutes measurement period.";
      }
      enum 30-minutes {
        value 7;
```

```
        description
          "30 minutes measurement period.";
      }
      enum hour {
        value 8;
        description
          "One hour measurement period.";
      }
    }
    description
      "Enumeration to indicate the measurement period.";
  }

  typedef percentile {
    type decimal64 {
      fraction-digits 2;
    }
    description
      "The nth percentile of a set of data is the
       value at which n percent of the data is below it.";
  }

  typedef query-type {
    type enumeration {
      enum target-prefix {
        value 1;
        description
          "Query based on target prefix.";
      }
      enum target-port {
        value 2;
        description
          "Query based on target port number.";
      }
      enum target-protocol {
        value 3;
        description
          "Query based on target protocol.";
      }
      enum target-fqdn {
        value 4;
        description
          "Query based on target FQDN.";
      }
      enum target-uri {
        value 5;
        description
          "Query based on target URI.";
      }
      enum target-alias {
        value 6;
        description
          "Query based on target alias.";
      }
      enum mid {
        value 7;
        description
```

```
          "Query based on mitigation identifier (mid).";
      }
      enum source-prefix {
        value 8;
        description
          "Query based on source prefix.";
      }
      enum source-port {
        value 9;
        description
          "Query based on source port number.";
      }
      enum source-icmp-type {
        value 10;
        description
          "Query based on ICMP type";
      }
      enum content {
        value 11;
        description
          "Query based on 'c' Uri-Query option that is used
           to control the selection of configuration
            and non-configuration data nodes.";
        reference
          "Section 4.4.2 of RFC SSSS.";  8782.";
      }
    }
    description
      "Enumeration support for query types that can be used
       in a GET request to filter out data.";
  }

  grouping percentile-config {
    description
      "Configuration of low, mid, and high percentile values.";
    leaf measurement-interval {
      type interval;
      description
        "Defines the period on which percentiles are computed.";
    }
    leaf measurement-sample {
      type sample;
      description
        "Defines the time distribution for measuring
         values that are used to compute percentiles.";
    }
    leaf low-percentile {
      type percentile;
      default "10.00";
      description
        "Low percentile. If set to '0', this means low-percentiles
         are disabled.";
    }
    leaf mid-percentile {
      type percentile;
      must '. >= ../low-percentile' {
        error-message
```

```
          "The mid-percentile must be greater than
           or equal to the low-percentile.";
      }
      default "50.00";
      description
        "Mid percentile. If set to the same value as low-percentiles,
         this means mid-percentiles are disabled.";
    }
    leaf high-percentile {
      type percentile;
      must '. >= ../mid-percentile' {
        error-message
          "The high-percentile must be greater than
           or equal to the mid-percentile.";
      }
      default "90.00";
      description
        "High percentile. If set to the same value as mid-percentiles,
         this means high-percentiles are disabled.";
    }
  }

  grouping percentile {
    description
      "Generic grouping for percentile.";
    leaf low-percentile-g {
      type yang:gauge64;
      description
        "Low percentile value.";
    }
    leaf mid-percentile-g {
      type yang:gauge64;
      description
        "Mid percentile value.";
    }
    leaf high-percentile-g {
      type yang:gauge64;
      description
        "High percentile value.";
    }
    leaf peak-g {
      type yang:gauge64;
      description
        "Peak value.";
    }
  }

  grouping unit-config {
    description
      "Generic grouping for unit configuration.";
    list unit-config {
      key "unit";
      description
        "Controls which unit types are allowed when sharing
         telemetry data.";
      leaf unit {
        type unit-type;
```

```
      description
        "Can be packet-ps, bit-ps, or byte-ps.";
    }
    leaf unit-status {
      type boolean;
      mandatory true;
      description
        "Enable/disable the use of the measurement unit type.";
    }
  }
}

grouping traffic-unit {
  description
    "Grouping of traffic as a function of the measurement unit.";
  leaf unit {
    type unit;
    description
      "The traffic can be measured using unit types: packet-ps,
       bit-ps, or byte-ps. DOTS agents auto-scale to the appropriate
       units (e.g., megabit-ps, kilobit-ps).";
  }
  uses percentile;
}

grouping traffic-unit-protocol {
  description
    "Grouping of traffic of a given transport protocol as
     a function of the measurement unit.";
  leaf protocol {
    type uint8;
    description
      "The transport protocol.
       Values are taken from the IANA Protocol Numbers registry:
       <https://www.iana.org/assignments/protocol-numbers/>.

       For example, this parameter contains 6 for TCP,
       17 for UDP, 33 for DCCP, or 132 for SCTP.";
  }
  uses traffic-unit;
}

grouping traffic-unit-port {
  description
    "Grouping of traffic bound to a port number as
     a function of the measurement unit.";
  leaf port {
    type inet:port-number;
    description
      "Port number.";
  }
  uses traffic-unit;
}

grouping total-connection-capacity {
  description
    "Total Connections Capacity. These attributes are
```

```
   useful to detect resource consuming DDoS attacks";
leaf connection {
  type uint64;
  description
    "The maximum number of simultaneous connections that
     are allowed to the target server.";
}
leaf connection-client {
  type uint64;
  description
    "The maximum number of simultaneous connections that
     are allowed to the target server per client.";
}
leaf embryonic {
  type uint64;
  description
    "The maximum number of simultaneous embryonic connections
     that are allowed to the target server. The term 'embryonic
     connection' refers to a connection whose connection handshake
     is not finished. Embryonic connection is only possible in
     connection-oriented transport protocols like TCP or SCTP.";
}
leaf embryonic-client {
  type uint64;
  description
    "The maximum number of simultaneous embryonic connections
     that are allowed to the target server per client.";
}
leaf connection-ps {
  type uint64;
  description
    "The maximum number of connections allowed per second
     to the target server.";
}
leaf connection-client-ps {
  type uint64;
  description
    "The maximum number of connections allowed per second
     to the target server per client.";
}
leaf request-ps {
  type uint64;
  description
    "The maximum number of requests allowed per second
     to the target server.";
}
leaf request-client-ps {
  type uint64;
  description
    "The maximum number of requests allowed per second
     to the target server per client.";
}
leaf partial-request-ps {
  type uint64;
  description
    "The maximum number of partial requests allowed per
     second to the target server.";
```

```
    }
    leaf partial-request-client-ps {
      type uint64;
      description
        "The maximum number of partial requests allowed per
         second to the target server per client.";
    }
  }

  grouping total-connection-capacity-protocol {
    description
      "Total Connections Capacity per protocol. These attributes are
       useful to detect resource consuming DDoS attacks.";
    leaf protocol {
      type uint8;
      description
        "The transport protocol.
         Values are taken from the IANA Protocol Numbers registry:
         <https://www.iana.org/assignments/protocol-numbers/>.";
    }
    uses total-connection-capacity;
  }

  grouping connection {
    description
      "A set of attributes which represent the attack
       characteristics";
    leaf connection {
      type yang:gauge64;
      description
        "The number of simultaneous attack connections to
         the target server.";
    }
    leaf embryonic {
      type yang:gauge64;
      description
        "The number of simultaneous embryonic connections to
         the target server.";
    }
    leaf connection-ps {
      type yang:gauge64;
      description
        "The number of attack connections per second to
         the target server.";
    }
    leaf request-ps {
      type yang:gauge64;
      description
        "The number of attack requests per second to
         the target server.";
    }
    leaf partial-request-ps {
      type yang:gauge64;
      description
        "The number of attack partial requests to
         the target server.";
    }
```

```
  }

  grouping connection-percentile {
    description
      "Total attack connections.";
    container low-percentile-c {
      description
        "Low percentile of attack connections.";
      uses connection;
    }
    container mid-percentile-c {
      description
        "Mid percentile of attack connections.";
      uses connection;
    }
    container high-percentile-c {
      description
        "High percentile of attack connections.";
      uses connection;
    }
    container peak-c {
      description
        "Peak attack connections.";
      uses connection;
    }
  }

  grouping connection-protocol {
    description
      "Total attack connections.";
    leaf protocol {
      type uint8;
      description
        "The transport protocol.
         Values are taken from the IANA Protocol Numbers registry:
         <https://www.iana.org/assignments/protocol-numbers/>.";
    }
    uses connection;
  }

  grouping connection-port {
    description
      "Total attack connections per port number.";
    leaf port {
      type inet:port-number;
      description
        "Port number.";
    }
    uses connection-protocol;
  }

  grouping connection-protocol-percentile {
    description
      "Total attack connections per protocol.";
    list low-percentile-l {
      key "protocol";
      description
```

```
        "Low percentile of attack connections per protocol.";
      uses connection-protocol;
    }
    list mid-percentile-l {
      key "protocol";
      description
        "Mid percentile of attack connections per protocol.";
      uses connection-protocol;
    }
    list high-percentile-l {
      key "protocol";
      description
        "High percentile of attack connections per protocol.";
      uses connection-protocol;
    }
    list peak-l {
      key "protocol";
      description
        "Peak attack connections per protocol.";
      uses connection-protocol;
    }
  }

  grouping connection-protocol-port-percentile {
    description
      "Total attack connections per port number.";
    list low-percentile-l {
      key "protocol port";
      description
        "Low percentile of attack connections per port number.";
      uses connection-port;
    }
    list mid-percentile-l {
      key "protocol port";
      description
        "Mid percentile of attack connections per port number.";
      uses connection-port;
    }
    list high-percentile-l {
      key "protocol port";
      description
        "High percentile of attack connections per port number.";
      uses connection-port;
    }
    list peak-l {
      key "protocol port";
      description
        "Peak attack connections per port number.";
      uses connection-port;
    }
  }

  grouping attack-detail {
    description
      "Various details that describe the on-going
       attacks that need to be mitigated by the DOTS server.
       The attack details need to cover well-known and common attacks
```

```
       (such as a SYN Flood) along with new emerging or vendor-specific
       attacks.";
    leaf vendor-id {
      type uint32;
      description
        "Vendor ID is a security vendor's Enterprise Number.";
    }
    leaf attack-id {
      type uint32;
      description
        "Unique identifier assigned by the vendor for the attack.";
    }
    leaf attack-description {
      type string;
      description
        "Textual representation of attack description. Natural Language
         Processing techniques (e.g., word embedding) can possibly be used
         to map the attack description to an attack type.";
    }
    leaf attack-severity {
      type attack-severity;
      description
        "Severity level of an attack. How this level is determined
         is implementation-specific.";
    }
    leaf start-time {
      type uint64;
      description
        "The time the attack started. Start time is represented in seconds
         relative to 1970-01-01T00:00:00Z in UTC time.";
    }
    leaf end-time {
      type uint64;
      description
        "The time the attack ended. End time is represented in seconds
         relative to 1970-01-01T00:00:00Z in UTC time.";
    }
    container source-count {
      description
        "Indicates the count of unique sources involved
         in the attack.";
      uses percentile;
    }
  }

  grouping top-talker-aggregate {
    description
      "Top attack sources.";
    list talker {
      key "source-prefix";
      description
        "IPv4 or IPv6 prefix identifying the attacker(s).";
      leaf spoofed-status {
        type boolean;
        description
          "Indicates whether this address is spoofed.";
      }
```

```
leaf source-prefix {
  type inet:ip-prefix;
  description
    "IPv4 or IPv6 prefix identifying the attacker(s).";
}
list source-port-range {
  key "lower-port";
  description
    "Port range. When only lower-port is
     present, it represents a single port number.";
  leaf lower-port {
    type inet:port-number;
    mandatory true;
    description
      "Lower port number of the port range.";
  }
  leaf upper-port {
    type inet:port-number;
    must '. >= ../lower-port' {
      error-message
        "The upper port number must be greater than
         or equal to lower port number.";
    }
    description
      "Upper port number of the port range.";
  }
}
list source-icmp-type-range {
  key "lower-type";
  description
    "ICMP type range. When only lower-type is
     present, it represents a single ICMP type.";
  leaf lower-type {
    type uint8;
    mandatory true;
    description
      "Lower ICMP type of the ICMP type range.";
  }
  leaf upper-type {
    type uint8;
    must '. >= ../lower-type' {
      error-message
        "The upper ICMP type must be greater than
         or equal to lower ICMP type.";
    }
    description
      "Upper type of the ICMP type range.";
  }
}
list total-attack-traffic {
  key "unit";
  description
    "Total attack traffic issued from this source.";
  uses traffic-unit;
}
container total-attack-connection {
  description
```

```
        "Total attack connections issued from this source.";
      uses connection-percentile;
    }
  }
}

grouping top-talker {
  description
    "Top attack sources.";
  list talker {
    key "source-prefix";
    description
      "IPv4 or IPv6 prefix identifying the attacker(s).";
    leaf spoofed-status {
      type boolean;
      description
        "Indicates whether this address is spoofed.";
    }
    leaf source-prefix {
      type inet:ip-prefix;
      description
        "IPv4 or IPv6 prefix identifying the attacker(s).";
    }
    list source-port-range {
      key "lower-port";
      description
        "Port range. When only lower-port is
         present, it represents a single port number.";
      leaf lower-port {
        type inet:port-number;
        mandatory true;
        description
          "Lower port number of the port range.";
      }
      leaf upper-port {
        type inet:port-number;
        must '. >= ../lower-port' {
          error-message
            "The upper port number must be greater than
             or equal to lower port number.";
        }
        description
          "Upper port number of the port range.";
      }
    }
    list source-icmp-type-range {
      key "lower-type";
      description
        "ICMP type range. When only lower-type is
         present, it represents a single ICMP type.";
      leaf lower-type {
        type uint8;
        mandatory true;
        description
          "Lower ICMP type of the ICMP type range.";
      }
      leaf upper-type {
```

```
        type uint8;
        must '. >= ../lower-type' {
          error-message
            "The upper ICMP type must be greater than
             or equal to lower ICMP type.";
        }
        description
          "Upper type of the ICMP type range.";
      }
    }
    list total-attack-traffic {
      key "unit";
      description
        "Total attack traffic issued from this source.";
      uses traffic-unit;
    }
    container total-attack-connection {
      description
        "Total attack connections issued from this source.";
      uses connection-protocol-percentile;
    }
  }
}

grouping baseline {
  description
    "Grouping for the telemetry baseline.";
  uses ietf-data:target;
  leaf-list alias-name {
    type string;
    description
      "An alias name that points to a resource.";
  }
  list total-traffic-normal {
    key "unit";
    description
      "Total traffic normal baselines.";
    uses traffic-unit;
  }
  list total-traffic-normal-per-protocol {
    key "unit protocol";
    description
      "Total traffic normal baselines per protocol.";
    uses traffic-unit-protocol;
  }
  list total-traffic-normal-per-port {
    key "unit port";
    description
      "Total traffic normal baselines per port number.";
    uses traffic-unit-port;
  }
  list total-connection-capacity {
    key "protocol";
    description
      "Total connection capacity.";
    uses total-connection-capacity-protocol;
  }
```

```
    list total-connection-capacity-per-port {
      key "protocol port";
      description
        "Total connection capacity per port number.";
      leaf port {
        type inet:port-number;
        description
          "The target port number.";
      }
      uses total-connection-capacity-protocol;
    }
  }

  grouping pre-or-ongoing-mitigation {
    description
      "Grouping for the telemetry data.";
    list total-traffic {
      key "unit";
      description
        "Total traffic.";
      uses traffic-unit;
    }
    list total-traffic-protocol {
      key "unit protocol";
      description
        "Total traffic per protocol.";
      uses traffic-unit-protocol;
    }
    list total-traffic-port {
      key "unit port";
      description
        "Total traffic per port.";
      uses traffic-unit-port;
    }
    list total-attack-traffic {
      key "unit";
      description
        "Total attack traffic.";
      uses traffic-unit-protocol;
    }
    list total-attack-traffic-protocol {
      key "unit protocol";
      description
        "Total attack traffic per protocol.";
      uses traffic-unit-protocol;
    }
    list total-attack-traffic-port {
      key "unit port";
      description
        "Total attack traffic per port.";
      uses traffic-unit-port;
    }
    container total-attack-connection {
      description
        "Total attack connections.";
      uses connection-protocol-percentile;
    }
```

```
    container total-attack-connection-port {
      description
        "Total attack connections.";
      uses connection-protocol-port-percentile;
    }
    list attack-detail {
      key "vendor-id attack-id";
      description
        "Provides a set of attack details.";
      uses attack-detail;
      container top-talker {
        description
          "Lists the top attack sources.";
        uses top-talker;
      }
    }
  }

  augment

  sx:augment-structure "/ietf-signal:dots-signal/ietf-signal:message-type/"
                     + "ietf-signal:mitigation-scope/ietf-signal:scope" {
    if-feature "dots-telemetry";
    description
      "Extends mitigation scope with telemetry update data.";
    choice direction {
      description
        "Indicates the communication direction in which the
         attributes can be included.";
      case server-to-client-only {
        description
          "These attributes appear only in a mitigation message
           sent from the server to the client.";
        list total-traffic {
          key "unit";
      config false;
          description
            "Total traffic.";
          uses traffic-unit;
        }
    list total-attack-traffic
        container total-attack-connection {
      key "unit";
          description
            "Total attack traffic."; connections.";
          uses traffic-unit; connection-percentile;
        }
    container total-attack-connection
      }
    }
    list total-attack-traffic {
      config false;
      key "unit";
      description
        "Total attack connections."; traffic.";
      uses connection-percentile; traffic-unit;
    }
```

```
      list attack-detail {
        key "vendor-id attack-id";
        description
          "Attack details";
        uses attack-detail;
        container top-talker {
          description
            "Top attack sources.";
          uses top-talker-aggregate;
        }
      }
    }

    augment "/ietf-signal:dots-signal/ietf-signal:message-type"
    sx:structure dots-telemetry {
      if-feature "dots-telemetry";
      description
        "Add a new
        "Main container for DOTS telemetry message.";
      choice to enclose telemetry-message-type {
        description
          "Can be a telemetry-setup or telemetry data in DOTS
          signal channel."; data.";
        case telemetry-setup {
          description
            "Indicates the message is about telemetry.";
          choice direction {
            description
              "Indicates the communication direction in which the
               attributes can be included.";
            case server-to-client-only {
              description
                "These attributes appear only in a mitigation message
                 sent from the server to the client.";
              container max-config-values {
          config false;
                description
                  "Maximum acceptable configuration values.";
                uses percentile-config;
                leaf server-originated-telemetry {
                  type boolean;
                  description
                    "Indicates whether the DOTS server can be instructed
                     to send pre-or-ongoing-mitigation telemetry. If set to FALSE
                     or the attribute is not present, this is an indication
                     that the server does not support this capability.";
                }
                leaf telemetry-notify-interval {
                  type uint32 {
                    range "1 .. 3600";
                  }
                  must '. >= ../../min-config-values/telemetry-notify-interval' {
                    error-message
                      "The value must be greater than or equal
                       to the telemetry-notify-interval in the min-config-values";
                  }
                  units "seconds";
```

```
        description
          "Minimum number of seconds between successive
           telemetry notifications.";
      }
    }
    container min-config-values {
~~config false;~~
      description
        "Minimum acceptable configuration values.";
      uses percentile-config;
      leaf telemetry-notify-interval {
        type uint32 {
          range "1 .. 3600";
        }
        units "seconds";
        description
          "Minimum number of seconds between successive
           telemetry notifications.";
      }
    }
    container supported-units {
~~config false;~~
      description
        "Supported units and default activation status.";
      uses unit-config;
    }
    leaf-list query-type {
      type query-type;
~~config false;~~
      description
        "Indicates which query types are supported by
         the server.";
    }
  }
}
list telemetry {
  key "cuid tsid";
  description
    "The telemetry data per DOTS client.";
  leaf cuid {
    type string;
    description
      "A unique identifier that is
       generated by a DOTS client to prevent
       request collisions.  It is expected that the
       cuid will remain consistent throughout the
       lifetime of the DOTS client.";
  }
  leaf cdid {
    type string;
    description
      "The cdid should be included by a server-domain
       DOTS gateway to propagate the client domain
       identification information from the
       gateway's client-facing-side to the gateway's
       server-facing-side, and from the gateway's
       server-facing-side to the DOTS server.
```

```
      It may be used by the final DOTS server
      for policy enforcement purposes.";
  }
  leaf tsid {
    type uint32;
    description
      "An identifier for the DOTS telemetry setup
       data.";
  }
  choice setup-type {
    description
      "Can be a mitigation configuration, a pipe capacity,
       or baseline message.";
    case telemetry-config {
      description
        "Uses to set low, mid, and high percentile values.";
      container current-config {
        description
          "Current configuration values.";
        uses percentile-config;
        uses unit-config;
        leaf server-originated-telemetry {
          type boolean;
          description
            "Used by a DOTS client to enable/disable whether it
             accepts pre-or-ongoing-mitigation telemetry from
             the DOTS server.";
        }
        leaf telemetry-notify-interval {
          type uint32 {
            range "1 .. 3600";
          }
          units "seconds";
          description
            "Minimum number of seconds between successive
             telemetry notifications.";
        }
      }
    }
    case pipe {
      description
        "Total pipe capacity of a DOTS client domain";
      list total-pipe-capacity {
        key "link-id unit";
        description
          "Total pipe capacity of a DOTS client domain.";
        leaf link-id {
          type nt:link-id;
          description
            "Identifier of an interconnection link.";
        }
        leaf capacity {
          type uint64;
          mandatory true;
          description
            "Pipe capacity.";
```

```
            }
            leaf unit {
              type unit;
              description
                "The traffic can be measured using unit types: packets
                 per second (PPS), Bits per Second (BPS), and/or
                 bytes per second. DOTS agents auto-scale to the
                 appropriate units (e.g., megabit-ps, kilobit-ps).";
            }
          }
        }
        case baseline {
          description
            "Traffic baseline information";
          list baseline {
            key "id";
            description
              "Traffic baseline information";
            leaf id {
              type uint32;
              must '. >= 1';
              description
                "A baseline entry identifier.";
            }
            uses baseline;
          }
        }
      }
    }
  }
  case telemetry {
    description
      "Indicates the message is about telemetry.";
    list pre-or-ongoing-mitigation {
      key "cuid tmid";
      description
        "Pre-or-ongoing-mitigation telemetry per DOTS client.";
      leaf cuid {
        type string;
        description
          "A unique identifier that is
           generated by a DOTS client to prevent
           request collisions.  It is expected that the
           cuid will remain consistent throughout the
           lifetime of the DOTS client.";
      }
      leaf cdid {
        type string;
        description
          "The cdid should be included by a server-domain
           DOTS gateway to propagate the client domain
           identification information from the
           gateway's client-facing-side to the gateway's
           server-facing-side, and from the gateway's
           server-facing-side to the DOTS server.

           It may be used by the final DOTS server
```

```
                for policy enforcement purposes.";
            }
            leaf tmid {
              type uint32;
              description
                "An identifier to uniquely demux telemetry data sent
                 using the same message.";
            }
            container target {
              description
                "Indicates the target.";
              uses ietf-data:target;
              leaf-list alias-name {
                type string;
                description
                  "An alias name that points to a resource.";
              }
              leaf-list mid-list {
                type uint32;
                description
                  "Reference a list of associated mitigation requests.";
              }
            }
            uses pre-or-ongoing-mitigation;
          }
        }
      }
    }
  }
```