DOTS
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2020

M. Boucadair, Ed.
Orange
T. Reddy, Ed.
McAfee
E. Doron
Radware Ltd.
M. Chen
CMCC
March 19, 2020

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry
draft-ietf-dots-telemetry-04

Abstract

This document aims to enrich DOTS signal channel protocol with various telemetry attributes allowing optimal DDoS attack mitigation. This document specifies the normal traffic baseline and attack traffic telemetry attributes a DOTS client can convey to its DOTS server in the mitigation request, the mitigation status telemetry attributes a DOTS server can communicate to a DOTS client, and the mitigation efficacy telemetry attributes a DOTS client can communicate to a DOTS server. The telemetry attributes can assist the mitigator to choose the DDoS mitigation techniques and perform optimal DDoS attack mitigation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Distributed Denial of Service (DDoS) attacks have become more vicious and sophisticated in almost all aspects of their maneuvers and malevolent intentions. IT organizations and service providers are facing DDoS attacks that fall into two broad categories: Network/Transport layer attacks and Application layer attacks:

- o Network/Transport layer attacks target the victim's infrastructure. These attacks are not necessarily aimed at taking down the actual delivered services, but rather to eliminate various network elements (routers, switches, firewalls, transit links, and so on) from serving legitimate user traffic.

  The main method of such attacks is to send a large volume or high PPS of traffic toward the victim's infrastructure. Typically, attack volumes may vary from a few 100 Mbps/PPS to 100s of Gbps or even Tbps. Attacks are commonly carried out leveraging botnets and attack reflectors for amplification attacks such as NTP (Network Time Protocol), DNS (Domain Name System), SNMP (Simple Network Management Protocol), or SSDP (Simple Service Discovery Protoco).

- Application layer attacks target various applications. Typical examples include attacks against HTTP/HTTPS, DNS, SIP (Session Initiation Protocol), or SMTP (Simple Mail Transfer Protocol). However, all valid applications with their port numbers open at network edges can be attractive attack targets.

  Application layer attacks are considered more complex and hard to categorize, therefore harder to detect and mitigate efficiently.

To compound the problem, attackers also leverage multi-vectored attacks. These attacks are assembled from dynamic attack vectors (Network/Application) and tactics. As such, multiple attack vectors formed by multiple attack types and volumes are launched simultaneously towards a victim. Multi-vector attacks are harder to detect and defend. Multiple and simultaneous mitigation techniques are needed to defeat such attack campaigns. It is also common for attackers to change attack vectors right after a successful mitigation, burdening their opponents with changing their defense methods.

The ultimate conclusion derived from these real scenarios is that modern attacks detection and mitigation are most certainly complicated and highly convoluted tasks. They demand a comprehensive knowledge of the attack attributes, the targeted normal behavior/ traffic patterns, as well as the attacker's on-going and past actions. Even more challenging, retrieving all the analytics needed for detecting these attacks is not simple to obtain with the industry's current capabilities.

The DOTS signal channel protocol [I-D.ietf-dots-signal-channel] is used to carry information about a network resource or a network (or a part thereof) that is under a DDoS attack. Such information is sent by a DOTS client to one or multiple DOTS servers so that appropriate mitigation actions are undertaken on traffic deemed suspicious. Various use cases are discussed in [I-D.ietf-dots-use-cases].

Typically, DOTS clients can be integrated within a DDoS attack detector, or network and security elements that have been actively engaged with ongoing attacks. The DOTS client mitigation environment determines that it is no longer possible or practical for it to handle these attacks. This can be due to lack of resources or security capabilities, as derived from the complexities and the intensity of these attacks. In this circumstance, the DOTS client has invaluable knowledge about the actual attacks that need to be handled by its DOTS server(s). By enabling the DOTS client to share this comprehensive knowledge of an ongoing attack under specific circumstances, the DOTS server can drastically increase its ability to accomplish successful mitigation. While the attack is being

handled by the DOTS server associated mitigation resources, the DOTS server has the knowledge about the ongoing attack mitigation. The DOTS server can share this information with the DOTS client so that the client can better assess and evaluate the actual mitigation realized.

In some deployments, DOTS clients can send mitigation hints derived from attack details to DOTS servers, with the full understanding that the DOTS server may ignore mitigation hints, as described in [RFC8612] (Gen-004). Mitigation hints will be transmitted across the DOTS signal channel, as the data channel may not be functional during an attack. How a DOTS server is handling normal and attack traffic attributes, and mitigation hints is implementation-specific.

Both DOTS client and server can benefit this information by presenting various information in relevant management, reporting, and portal systems.

This document defines DOTS telemetry attributes the DOTS client can convey to the DOTS server, and vice versa. The DOTS telemetry attributes are not mandatory fields. Nevertheless, when DOTS telemetry attributes are available to a DOTS agent, and absent any policy, it can signal the attributes in order to optimize the overall mitigation service provisioned using DOTS. Some of the DOTS telemetry data is not shared during an attack time.

2. Terminology

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

The reader should be familiar with the terms defined in [RFC8612].

"DOTS Telemetry" is defined as the collection of attributes that are used to characterize normal traffic baseline, attacks and their mitigation measures, and any related information that may help in enforcing countermeasures. The DOTS Telemetry is an optional set of attributes that can be signaled in the DOTS signal channel protocol.

The meaning of the symbols in YANG tree diagrams is defined in [RFC8340].

3. DOTS Telemetry: Overview and Purpose

When signaling a mitigation request, it is most certainly beneficial for the DOTS client to signal to the DOTS server any knowledge regarding ongoing attacks. This can happen in cases where DOTS clients are asking the DOTS server for support in defending against attacks that they have already detected and/or mitigated. These actions taken by DOTS clients are referred to as "signaling the DOTS Telemetry".

If attacks are already detected and categorized by the DOTS client domain, the DOTS server, and its associated mitigation services, can proactively benefit this information and optimize the overall service delivered. It is important to note that DOTS client and server detection and mitigation approaches can be different, and can potentially outcome different results and attack classifications. The DDoS mitigation service treats the ongoing attack details from the client as hints and cannot completely rely or trust the attack details conveyed by the DOTS client.

A basic requirement of security operation teams is to be aware and get visibility into the attacks they need to handle. The DOTS server security operation teams benefit from the DOTS telemetry, especially from the reports of ongoing attacks. Even if some mitigation can be automated, operational teams can use the DOTS telemetry to be prepared for attack mitigation and to assign the correct resources (operation staff, networking and mitigation) for the specific service. Similarly, security operation personnel at the DOTS client side ask for feedback about their requests for protection. Therefore, it is valuable for the DOTS server to share DOTS telemetry with the DOTS client.

Thus mutual sharing of information is crucial for "closing the mitigation loop" between the DOTS client and server. For the server side team, it is important to realize that the same attacks that the DOTS server's mitigation resources are seeing are those that the DOTS client is asking to mitigate. For the DOTS client side team, it is important to realize that the DOTS clients receive the required service. For example, understanding that "I asked for mitigation of two attacks and my DOTS server detects and mitigates only one...". Cases of inconsistency in attack classification between DOTS client and server can be high-lighted, and maybe handled, using the DOTS telemetry attributes.

In addition, management and orchestration systems, at both DOTS client and server sides, can potentially use DOTS telemetry as a feedback to automate various control and management activities derived from ongoing information signaled.

If the DOTS server's mitigation resources have the capabilities to facilitate the DOTS telemetry, the DOTS server adopts its protection strategy and activates the required countermeasures immediately (automation enabled). The overall results of this adoption are optimized attack mitigation decisions and actions.

The DOTS telemetry can also be used to tune the DDoS mitigators with the correct state of the attack. During the last few years, DDoS attack detection technologies have evolved from threshold-based detection (that is, cases when all or specific parts of traffic cross a pre-defined threshold for a certain period of time is considered as an attack) to an "anomaly detection" approach. In anomaly detection, the main idea is to maintain rigorous learning of "normal" behavior and where an "anomaly" (or an attack) is identified and categorized based on the knowledge about the normal behavior and a deviation from this normal behavior. Machine learning approaches are used such that the actual "traffic thresholds" are "automatically calculated" by learning the protected entity normal traffic behavior during peace time. The normal traffic characterization learned is referred to as the "normal traffic baseline". An attack is detected when the victim's actual traffic is deviating from this normal baseline.

In addition, subsequent activities toward mitigating an attack are much more challenging. The ability to distinguish legitimate traffic from attacker traffic on a per packet basis is complex. This complexity originates from the fact that the packet itself may look "legitimate" and no attack signature can be identified. The anomaly can be identified only after detailed statistical analysis. DDoS attack mitigators use the normal baseline during the mitigation of an attack to identify and categorize the expected appearance of a specific traffic pattern. Particularly the mitigators use the normal baseline to recognize the "level of normality" needs to be achieved during the various mitigation process.

Normal baseline calculation is performed based on continuous learning of the normal behavior of the protected entities. The minimum learning period varies from hours to days and even weeks, depending on the protected application behavior. The baseline cannot be learned during active attacks because attack conditions do not characterize the protected entities' normal behavior.

If the DOTS client has calculated the normal baseline of its protected entities, signaling this attribute to the DOTS server along with the attack traffic levels is significantly valuable. The DOTS server benefits from this telemetry by tuning its mitigation resources with the DOTS client's normal baseline. The DOTS server mitigators use the baseline to familiarize themselves with the attack victim's normal behavior and target the baseline as the level of

normality they need to achieve. Consequently, the overall mitigation performances obtained are dramatically improved in terms of time to mitigate, accuracy, false-negative, false-positive, and other measures.

Mitigation of attacks without having certain knowledge of normal traffic can be inaccurate at best. This is especially true for recursive signaling (see Section 3.2.3 in [I-D.ietf-dots-use-cases]). In addition, the highly diverse types of use-cases where DOTS clients are integrated also emphasize the need for knowledge of client behavior. Consequently, common global thresholds for attack detection practically cannot be realized. Each DOTS client can have its own levels of traffic and normal behavior. Without facilitating normal baseline signaling, it may be very difficult for DOTS servers in some cases to detect and mitigate the attacks accurately:

> It is important to emphasize that it is practically impossible for the server's mitigators to calculate the normal baseline in cases where they do not have any knowledge of the traffic beforehand.
>
> In addition, baseline learning requires a period of time that cannot be afforded during active attack.
>
> Of course, this information can provided using out-of-band mechanisms or manual configuration at the risk to maintain inaccurate information as the network evolves and "normal" patterns change. The use of a dynamic and collaborative means between the DOTS client and server to identify and share key parameters for the sake of efficient DDoS protection is valuable.

During a high volume attack, DOTS client pipes can be totally saturated. The DOTS client asks the DOTS server to handle the attack upstream so that DOTS client pipes return to a reasonable load level (normal pattern, ideally). At this point, it is essential to ensure that the mitigator does not overwhelm the DOTS client pipes by sending back "clean traffic", or what it believes is "clean". This can happen when the mitigator has not managed to detect and mitigate all the attacks launched towards the client. In this case, it can be valuable to clients to signal to server the "Total pipe capacity", which is the level of traffic the DOTS client domain can absorb from the upstream network. Dynamic updates of the condition of pipes between DOTS agents while they are under a DDoS attack is essential. For example, where multiple DOTS clients share the same physical connectivity pipes. It is important to note, that the term "pipe" noted here does not necessary represent physical pipe, but rather represents the maximum level of traffic that the DOTS client domain can receive. The DOTS server should activate other mechanisms to ensure it does not allow the client's pipes to be saturated

unintentionally. The rate-limit action defined in [I-D.ietf-dots-data-channel] is a reasonable candidate to achieve this objective; the client can ask for the type of traffic (such as ICMP, UDP, TCP port number 80) it prefers to limit. The rate-limit action can be controlled via the signal-channel [I-D.ietf-dots-signal-filter-control] even when the pipe is overwhelmed.

To summarize:

> Timely and effective signaling of up-to-date DOTS telemetry to all elements involved in the mitigation process is essential and absolutely improves the overall service effectiveness. Bi-directional feedback between DOTS agents is required for the increased awareness of each party, supporting superior and highly efficient attack mitigation service.

4. Generic Considerations

4.1. DOTS Client Identification

Following the rules in [I-D.ietf-dots-signal-channel], a unique identifier is generated by a DOTS client to prevent request collisions ('cuid').

4.2. DOTS Gateways

DOTS gateways may be located between DOTS clients and servers. The considerations elaborated in [I-D.ietf-dots-signal-channel] must be followed. In particular, 'cdid' attribute is used to unambiguously identify a DOTS client domain.

4.3. Empty URI Paths

Uri-Path parameters and attributes with empty values MUST NOT be present in a request and render an entire message invalid.

4.4. Controlling Configuration Data

The DOTS server follows the same considerations discussed in Section of 4.5.3 of [I-D.ietf-dots-signal-channel] for managing DOTS telemetry configuration freshness and notification. Likewise, a DOTS client may control the selection of configuration and non-configuration data nodes when sending a GET request by means of the 'c' Uri-Query option and following the procedure specified in Section of 4.4.2 of [I-D.ietf-dots-signal-channel]. These considerations are not re-iterated in the following sections.

4.5. Block-wise Transfer

DOTS clients can use Block-wise transfer [RFC7959] with the recommendation detailed in Section 4.4.2 of [I-D.ietf-dots-signal-channel] to control the size of a response when the data to be returned does not fit within a single datagram.

DOTS clients can also use Block1 Option in a PUT request (see Section 2.5 of [RFC7959]) to initiate large transfers, but these Block1 transfers will fail if the inbound "pipe" is running full, so consideration needs to be made to try to fit this PUT into a single transfer, or to separate out the PUT into several discrete PUTs where each of them fits into a single packet.

4.6. DOTS Multi-homing Considerations

Multi-homed DOTS clients are assumed to follow the recommendations in [I-D.ietf-dots-multihoming] to select which DOTS server to contact and which IP prefixes to include in a telemetry message to a given peer DOTS server. For example, if each upstream network exposes a DOTS server and the DOTS client maintains DOTS channels with all of them, only the information related to prefixes assigned by an upstream network to the DOTS client domain will be signaled via the DOTS channel established with the DOTS server of that upstream network. Considerations related to whether (and how) a DOTS client gleans some telemetry information (e.g., attack details) it receives from a first DOTS server and share it with a second DOTS server are implementation- and deployment-specific.

4.7. YANG Considerations

Messages exchanged between DOTS agents are serialized using Concise Binary Object Representation (CBOR). CBOR-encoded payloads are used to carry signal channel-specific payload messages which convey request parameters and response information such as errors [I-D.ietf-dots-signal-channel].

This document specifies a YANG module for representing DOTS telemetry message types (Section 9). All parameters in the payload of the DOTS signal channel are mapped to CBOR types as specified in Section 10.

The DOTS telemetry module (Section 9) uses "enumerations" rather than "identities" to define units, samples, and intervals because otherwise the namespace identifier "ietf-dots-telemetry" must be included when a telemetry attribute is included (e.g., in a mitigation efficacy update). The use of "identities" is thus suboptimal from a message compactness standpoint.

4.8. A Note About Examples

Examples are provided for illustration purposes. The document does not aim to provide a comprehensive list of message examples.

The authoritative reference for validating telemetry messages is the YANG module (Section 9) and the mapping table established in Section 10.

5. Telemetry Operation Paths

As discussed in [I-D.ietf-dots-signal-channel], each DOTS operation is indicated by a path-suffix that indicates the intended operation. The operation path is appended to the path-prefix to form the URI used with a CoAP request to perform the desired DOTS operation. The following telemetry path-suffixes are defined (Table 1):

Operation	Operation Path	Details
Telemetry Setup	/tm-setup	Section 6
Telemetry	/tm	Section 7

Table 1: DOTS Telemetry Operations

Consequently, the "ietf-dots-telemetry" YANG module defined in this document (Section 9) augments the "ietf-dots-signal" with two new message types called "telemetry-setup" and "telemetry". The tree structure is shown in Figure 1 (more details are provided in the following sections about the exact structure of "telemetry-setup" and "telemetry" message types).

```
  augment /ietf-signal:dots-signal/ietf-signal:message-type:
    +--:(telemetry-setup) {dots-telemetry}?
    |   ...
    |     +--rw (setup-type)?
    |        +--:(telemetry-config)
    |        |  ...
    |        +--:(pipe)
    |        | ...
    |        +--:(baseline)
    |        ...
    +--:(telemetry) {dots-telemetry}?
       ...
```

Figure 1: New DOTS Message Types (YANG Tree Structure)

6. DOTS Telemetry Setup Configuration

In reference to Figure 1, a DOTS telemetry setup message MUST include only telemetry-related configuration parameters (Section 6.1) or information about DOTS client domain pipe capacity (Section 6.2) or telemetry traffic baseline (Section 6.3). As such, requests that include a mix of telemetry configuration, pipe capacity, or traffic baseline MUST be rejected by DOTS servers with a 4.00 (Bad Request).

A DOTS client can reset all installed DOTS telemetry setup configuration data following the considerations detailed in Section 6.4.

A DOTS server may detect conflicts when processing requests related to DOTS client domain pipe capacity or telemetry traffic baseline with requests from other DOTS clients of the same DOTS client domain. More details are included in Section 6.5.

DOTS telemetry setup configuration request and response messages are marked as Confirmable messages.

6.1. Telemetry Configuration

A DOTS client can negotiate with its server(s) a set of telemetry configuration parameters to be used for telemetry. Such parameters include:

o Percentile-related measurement parameters

o Measurement units

o Acceptable percentile values

o Telemetry notification interval

o Acceptable Server-originated telemetry

Section 11.3 of [RFC2330] includes more details about computing percentiles.

6.1.1. Retrieve Current DOTS Telemetry Configuration

A GET request is used to obtain acceptable and current telemetry configuration parameters on the DOTS server. This request may include a 'cdid' Path-URI when the request is relayed by a DOTS gateway. An example of such request is depicted in Figure 2.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

Figure 2: GET to Retrieve Current and Acceptable DOTS Telemetry Configuration

Upon receipt of such request, the DOTS server replies with a 2.05 (Content) response that conveys the current and telemetry parameters acceptable by the DOTS server. The tree structure of the response message body is provided in Figure 3. Note that the response also includes any pipe (Section 6.2) and baseline information (Section 6.3) maintained by the DOTS server for this DOTS client.

DOTS servers that support the capability of sending telemetry information to DOTS clients prior or during a mitigation (Section 8.2) sets 'server-originated-telemetry' under 'max-config-values' to 'true' ('false' is used otherwise). If 'server-originated-telemetry' is not present in a response, this is equivalent to receiving a request with 'server-originated-telemetry'' set to 'false'.

```
     augment /ietf-signal:dots-signal/ietf-signal:message-type:
       +--:(telemetry-setup) {dots-telemetry}?
       |  +--rw telemetry* [cuid tsid]
       |     ...
       |     +--rw (setup-type)?
       |        +--:(telemetry-config)
       |        |  +--rw current-config
       |        |  |  +--rw measurement-interval?         interval
       |        |  |  +--rw measurement-sample?           sample
       |        |  |  +--rw low-percentile?               percentile
       |        |  |  +--rw mid-percentile?               percentile
       |        |  |  +--rw high-percentile?              percentile
       |        |  |  +--rw unit-config* [unit]
       |        |  |  |  +--rw unit           unit
       |        |  |  |  +--rw unit-status?   boolean
       |        |  |  +--rw server-originated-telemetry?   boolean
       |        |  |  +--rw telemetry-notify-interval?    uint32
       |        |  +--ro max-config-values
       |        |  |  +--ro measurement-interval?         interval
       |        |  |  +--ro measurement-sample?           sample
       |        |  |  +--ro low-percentile?               percentile
       |        |  |  +--ro mid-percentile?               percentile
       |        |  |  +--ro high-percentile?              percentile
       |        |  |  +--ro server-originated-telemetry?   boolean
       |        |  |  +--ro telemetry-notify-interval?    uint32
       |        |  +--ro min-config-values
       |        |  |  +--ro measurement-interval?        interval
       |        |  |  +--ro measurement-sample?          sample
       |        |  |  +--ro low-percentile?              percentile
       |        |  |  +--ro mid-percentile?              percentile
       |        |  |  +--ro high-percentile?             percentile
       |        |  |  +--ro telemetry-notify-interval?   uint32
       |        |  +--ro supported-units
       |        |     +--ro unit-config* [unit]
       |        |        +--ro unit           unit
       |        |        +--ro unit-status?   boolean
       |        +--:(pipe)
       |        ...
       |        +--:(baseline)
       |        ...
       +--:(telemetry) {dots-telemetry}?
          +--rw pre-or-ongoing-mitigation* [cuid tmid]
             ...
```

Figure 3: Telemetry Configuration Tree Structure

6.1.2. Convey DOTS Telemetry Configuration

PUT request is used to convey the configuration parameters for the telemetry data (e.g., low, mid, or high percentile values). For example, a DOTS client may contact its DOTS server to change the default percentile values used as baseline for telemetry data. Figure 3 lists the attributes that can be set by a DOTS client in such PUT request. An example of a DOTS client that modifies all percentile reference values is shown in Figure 4.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=123"
Content-Format: "application/dots+cbor"
```


```
{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
       "current-config": {
         "low-percentile": 5.00,
         "mid-percentile": 65.00,
         "high-percentile": 95.00
       }
     }
    ]
  }
}
```


Figure 4: PUT to Convey the DOTS Telemetry Configuration

'cuid' is a mandatory Uri-Path parameter for PUT requests.

The following additional Uri-Path parameter is defined:

tsid: Telemetry Setup Identifier is an identifier for the DOTS telemetry setup configuration data represented as an integer. This identifier MUST be generated by DOTS clients. 'tsid' values MUST increase monotonically (when a new PUT is generated by a DOTS client to convey new configuration parameters for the telemetry).

This is a mandatory attribute.

At least one configurable attribute MUST be present in the PUT request.

The PUT request with a higher numeric 'tsid' value overrides the DOTS telemetry configuration data installed by a PUT request with a lower numeric 'tsid' value. To avoid maintaining a long list of 'tsid' requests for requests carrying telemetry configuration data from a DOTS client, the lower numeric 'tsid' MUST be automatically deleted and no longer be available at the DOTS server.

The DOTS server indicates the result of processing the PUT request using the following response codes:

o If the request is missing a mandatory attribute, does not include 'cuid' or 'tsid' Uri-Path parameters, or contains one or more invalid or unknown parameters, 4.00 (Bad Request) MUST be returned in the response.

o If the DOTS server does not find the 'tsid' parameter value conveyed in the PUT request in its configuration data and if the DOTS server has accepted the configuration parameters, then a response code 2.01 (Created) MUST be returned in the response.

o If the DOTS server finds the 'tsid' parameter value conveyed in the PUT request in its configuration data and if the DOTS server has accepted the updated configuration parameters, 2.04 (Changed) MUST be returned in the response.

o If any of the enclosed configurable attribute values are not acceptable to the DOTS server (Section 6.1.1), 4.22 (Unprocessable Entity) MUST be returned in the response.

  The DOTS client may re-try and send the PUT request with updated attribute values acceptable to the DOTS server.

By default, low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile), and peak (100th percentile) values are used to represent telemetry data. Nevertheless, a DOTS client can disable some percentile types (low, mid, high). In particular, setting 'low-percentile' to '0.00' indicates that the DOTS client is not interested in receiving low-percentiles. Likewise, setting 'mid-percentile' (or 'high-percentile') to the same value as 'low-percentile' (or 'mid-percentile') indicates that the DOTS client is not interested in receiving mid-percentiles (or high-percentiles). For example, a DOTS client can send the request depicted in Figure 5 to inform the server that it is interested in receiving only high-percentiles. This

assumes that the client will only use that percentile type when sharing telemetry data with the server.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=569"
Content-Format: "application/dots+cbor"
```


```
{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
       "current-config": {
         "low-percentile": 0.00,
         "mid-percentile": 0.00,
         "high-percentile": 95.00
       }
     }
    ]
  }
}
```


Figure 5: PUT to Disable Low- and Mid-Percentiles

DOTS clients can also configure the unit(s) to be used for traffic-related telemetry data. Typically, the supported units are: packets per second (PPS) or kilo packets per second (Kpps) and Bits per Second (BPS), and kilobytes per second or megabytes per second or gigabytes per second.

DOTS clients that are interested to receive pre- or onoing mitigation telemetry (pre-or-ongoing-mitigation) information from a DOTS server (Section 8.2) MUST set 'server-originated-telemetry' to 'true'. If 'server-originated-telemetry' is not present in a PUT request, this is equivalent to receiving a request with 'server-originated-telemetry'' set to 'false'. An example of a request to enable pre-or-ongoing-mitigation telemetry from DOTS servers is shown in Figure 6.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=569"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
       "current-config": {
         "server-originated-telemetry": true
       }
     }
    ]
  }
}
```

Figure 6: PUT to Enable Pre-or-ongoing-mitigation Telemetry from the DOTS server

6.1.3. Retrieve Installed DOTS Telemetry Configuration

A DOTS client may issue a GET message with 'tsid' Uri-Path parameter to retrieve the current DOTS telemetry configuration. An example of such request is depicted in Figure 7.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=123"
```

Figure 7: GET to Retrieve Current DOTS Telemetry Configuration

If the DOTS server does not find the 'tsid' Uri-Path value conveyed in the GET request in its configuration data for the requesting DOTS client, it MUST respond with a 4.04 (Not Found) error response code.

6.1.4. Delete DOTS Telemetry Configuration

A DELETE request is used to delete the installed DOTS telemetry configuration data (Figure 8). 'cuid' and 'tsid' are mandatory Uri-Path parameters for such DELETE requests.

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=123"
```

Figure 8: Delete Telemetry Configuration

The DOTS server resets the DOTS telemetry configuration back to the default values and acknowledges a DOTS client's request to remove the DOTS telemetry configuration using 2.02 (Deleted) response code. A 2.02 (Deleted) Response Code is returned even if the 'tsid' parameter value conveyed in the DELETE request does not exist in its configuration data before the request.

Section 6.4 discusses the procedure to reset all DOTS telemetry setup configuration.

6.2. Total Pipe Capacity

A DOTS client can communicate to its server(s) its DOTS client domain pipe information. The tree structure of the pipe information is shown in Figure 9.

```
augment /ietf-signal:dots-signal/ietf-signal:message-type:
  +--:(telemetry-setup) {dots-telemetry}?
  |  +--rw telemetry* [cuid tsid]
  |     +--rw cuid                          string
  |     +--rw cdid?                         string
  |     +--rw tsid                          uint32
  |     +--rw (setup-type)?
  |        +--:(telemetry-config)
  |        |  ...
  |        +--:(pipe)
  |        |  +--rw total-pipe-capacity* [link-id unit]
  |        |     +--rw link-id     nt:link-id
  |        |     +--rw capacity    uint64
  |        |     +--rw unit        unit
  |        +--:(baseline)
  |        ...
  +--:(telemetry) {dots-telemetry}?
     +--rw pre-or-ongoing-mitigation* [cuid tmid]
        ...
```

Figure 9: Pipe Tree Structure

A DOTS client domain pipe is defined as a list of limits of (incoming) traffic volume (total-pipe-capacity") that can be forwarded over ingress interconnection links of a DOTS client domain. Each of these links is identified with a "link-id" [RFC8345].

This limit can be expressed in packets per second (PPS) or kilo packets per second (Kpps) and Bits per Second (BPS), and in kilobytes per second or megabytes per second or gigabytes per second. The unit used by a DOTS client when conveying pipe information is captured in 'unit' attribute.

6.2.1. Convey DOTS Client Domain Pipe Capacity

Similar considerations to those specified in Section 6.1.2 are followed with one exception:

> The relative order of two PUT requests carrying DOTS client domain pipe attributes from a DOTS client is determined by comparing their respective 'tsid' values. If such two requests have overlapping "link-id" and "unit", the PUT request with higher numeric 'tsid' value will override the request with a lower numeric 'tsid' value. The overlapped lower numeric 'tsid' MUST be automatically deleted and no longer be available.

DOTS clients SHOULD minimize the number of active 'tsids' used for pipe information. Typically, in order to avoid maintaining a long list of 'tsids' for pipe information, it is RECOMMENDED that DOTS clients include in any request to update information related to a given link the information of other links (already communicated using a lower 'tsid' value). Doing so, this update request will override these existing requests and hence optimize the number of 'tsid' request per DOTS client.

- o Note: This assumes that all link information can fit in one single message.

For example, a DOTS client managing a single homed domain (Figure 10) can send a PUT request (shown in Figure 11) to communicate the capacity of "link1" used to connect to its ISP.

```
       ,--,--,--.             ,--,--,--.
    ,-'          `-.       ,-'          `-.
   (  DOTS Client   )=====(     ISP#A      )
    `-.  Domain  ,-' link1 `-.          ,-'
       `--'--'--'             `--'--'--'
```

Figure 10: Single Homed DOTS Client Domain

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=457"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
      "total-pipe-capacity": [
        {
          "link-id": "link1",
          "capacity": 500,
          "unit": "megabytes-ps"
        }
      ]
     }
    ]
  }
}
```

Figure 11: Example of a PUT Request to Convey Pipe Information (Single Homed)

DOTS clients may be instructed to signal a link aggregate instead of individual links. For example, a DOTS client managing a DOTS client domain having two interconnection links with an upstream ISP (Figure 12) can send a PUT request (shown in Figure 13) to communicate the aggregate link capacity with its ISP. Signalling individual or aggregate link capacity is deployment-specific.

```
     ,--,--,--.             ,--,--,--.
  ,-'          `-.===== ,-'          `-.
 (  DOTS Client   )    (      ISP#C      )
  `-.  Domain  ,-'====== `-.          ,-'
     `--'--'--'             `--'--'--'
```

Figure 12: DOTS Client Domain with Two Interconnection Links

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=896"
Content-Format: "application/dots+cbor"
```


```
{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
      "total-pipe-capacity": [
        {
          "link-id": "aggregate",
          "capacity": 700,
          "unit": "megabytes-ps"
        }
      ]
     }
    ]
  }
}
```


Figure 13: Example of a PUT Request to Convey Pipe Information (Aggregated Link)

Now consider that the DOTS client domain was upgraded to connect to an additional ISP (ISP#B of Figure 14), the DOTS client can inform a third-party DOTS server (that is, not hosted with ISP#A and ISP#B domains) about this update by sending the PUT request depicted in Figure 15. This request also includes information related to "link1" even if that link is not upgraded. Upon receipt of this request, the DOTS server removes the request with 'tsid=457' and updates its configuration base to maintain two links (link#1 and link#2).

```
                   ,--,--,--.
                 ,-'          `-.
                (     ISP#B      )
                 `-.          ,-'
                    `--'--'--'
                        ||
                        || link2
                   ,--,--,--.             ,--,--,--.
                 ,-'          `-.       ,-'          `-.
                (  DOTS Client   )=====(     ISP#A      )
                 `-.  Domain  ,-' link1 `-.          ,-'
                    `--'--'--'             `--'--'--'
```

Figure 14: Multi-Homed DOTS Client Domain

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=458"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
      "total-pipe-capacity": [
        {
          "link-id": "link1",
          "capacity": 500,
          "unit": "megabytes-ps"
        },
        {
          "link-id": "link2",
          "capacity": 500,
          "unit": "megabytes-ps"
        }
      ]
     }
    ]
  }
}
```

Figure 15: Example of a PUT Request to Convey Pipe Information (Multi-Homed)

A DOTS client can delete a link by sending a PUT request with the 'capacity' attribute set to "0" if other links are still active for the same DOTS client domain (see Section 6.2.3 for other delete cases). For example, if a DOTS client domain re-homes (that is, it changes its ISP), the DOTS client can inform its DOTS server about this update (e.g., from the network configuration in Figure 10 to the one shown in Figure 16) by sending the PUT request depicted in Figure 17. Upon receipt of this request, the DOTS server removes "link1" from its configuration bases for this DOTS client domain.

```
                         ,--,--,--.
                       ,-'         `-.
                      (     ISP#B     )
                       `-.         ,-'
                          `--'--'--'
                              ||
                              || link2
                         ,--,--,--.
                       ,-'         `-.
                      (  DOTS Client  )
                       `-.  Domain ,-'
                          `--'--'--'
```

Figure 16: Multi-Homed DOTS Client Domain

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=459"
Content-Format: "application/dots+cbor"
```


```
{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
     {
      "total-pipe-capacity": [
        {
          "link-id": "link1",
          "capacity": 0,
          "unit": "megabytes-ps"
        },
        {
          "link-id": "link2",
          "capacity": 500,
          "unit": "megabytes-ps"
        }
      ]
     }
    ]
  }
}
```


Figure 17: Example of a PUT Request to Convey Pipe Information (Multi-Homed)

6.2.2. Retrieve Installed DOTS Client Domain Pipe Capacity

A GET request with 'tsid' Uri-Path parameter is used to retrieve a specific installed DOTS client domain pipe related information. The same procedure as defined in (Section 6.1.3) is followed.

To retrieve all pipe information bound to a DOTS client, the DOTS client proceeds as specified in Section 6.1.1.

6.2.3. Delete Installed DOTS Client Domain Pipe Capacity

A DELETE request is used to delete the installed DOTS client domain pipe related information. The same procedure as defined in (Section 6.1.4) is followed.

6.3. Telemetry Baseline

A DOTS client can communicate to its server(s) its normal traffic baseline and total connections capacity:

Total Traffic Normal Baseline: The percentile values representing the total traffic normal baseline.

The traffic normal baseline is represented for a target and is transport-protocol specific.

If the DOTS client negotiated percentile values and units (Section 6.1), these negotiated values will be used instead of the default ones.

Total Connections Capacity: If the target is subjected to resource consuming DDoS attacks, the following optional attributes for the target per transport-protocol are useful to detect resource consuming DDoS attacks:

Total Connections Capacity:

* The maximum number of simultaneous connections that are allowed to the target.
* The maximum number of simultaneous connections that are allowed to the target per client.
* The maximum number of simultaneous embryonic connections that are allowed to the target. The term "embryonic connection" refers to a connection whose connection handshake is not finished and embryonic connection is only possible in connection-oriented transport protocols like TCP or SCTP.
* The maximum number of simultaneous embryonic connections that are allowed to the target per client.
* The maximum number of connections allowed per second to the target.
* The maximum number of connections allowed per second to the target per client.
* The maximum number of requests allowed per second to the target.
* The maximum number of requests allowed per second to the target per client.
* The maximum number of partial requests allowed per second to the target.
* The maximum number of partial requests allowed per second to the target per client.

The threshold is transport-protocol.

The tree structure of the baseline is shown in Figure 18.

```
  augment /ietf-signal:dots-signal/ietf-signal:message-type:
    +--:(telemetry-setup) {dots-telemetry}?
    |  +--rw telemetry* [cuid tsid]
    |     +--rw cuid                         string
    |     +--rw cdid?                        string
    |     +--rw tsid                         uint32
    |     +--rw (setup-type)?
    |        +--:(telemetry-config)
    |        |  ...
    |        +--:(pipe)
    |        |  ...
    |        +--:(baseline)
    |           +--rw baseline* [id]
    |              +--rw id                                 uint32
    |              +--rw target-prefix*                     inet:ip-prefix
    |              +--rw target-port-range* [lower-port]
    |              |  +--rw lower-port    inet:port-number
    |              |  +--rw upper-port?   inet:port-number
    |              +--rw target-protocol*                   uint8
    |              +--rw target-fqdn*                       inet:domain-name
    |              +--rw target-uri*                        inet:uri
    |              +--rw total-traffic-normal-baseline* [unit protocol]
    |              |  +--rw unit                  unit
    |              |  +--rw protocol              uint8
    |              |  +--rw low-percentile-g?     yang:gauge64
    |              |  +--rw mid-percentile-g?     yang:gauge64
    |              |  +--rw high-percentile-g?    yang:gauge64
    |              |  +--rw peak-g?               yang:gauge64
    |              +--rw total-connection-capacity* [protocol]
    |                 +--rw protocol                      uint8
    |                 +--rw connection?                   uint64
    |                 +--rw connection-client?            uint64
    |                 +--rw embryonic?                    uint64
    |                 +--rw embryonic-client?             uint64
    |                 +--rw connection-ps?                uint64
    |                 +--rw connection-client-ps?         uint64
    |                 +--rw request-ps?                   uint64
    |                 +--rw request-client-ps?            uint64
    |                 +--rw partial-request-ps?           uint64
    |                 +--rw partial-request-client-ps?    uint64
    +--:(telemetry) {dots-telemetry}?
       +--rw pre-or-ongoing-mitigation* [cuid tmid]
          ...
```

Figure 18: Telemetry Baseline Tree Structure

6.3.1. Convey DOTS Client Domain Baseline Information

Similar considerations to those specified in Section 6.1.2 are followed with one exception:

> The relative order of two PUT requests carrying DOTS client domain baseline attributes from a DOTS client is determined by comparing their respective 'tsid' values. If such two requests have overlapping targets, the PUT request with higher numeric 'tsid' value will override the request with a lower numeric 'tsid' value. The overlapped lower numeric 'tsid' MUST be automatically deleted and no longer be available.

Two PUT requests from a DOTS client have overlapping targets if there is a common IP address, IP prefix, FQDN, or URI.

DOTS clients SHOULD minimize the number of active 'tsids' used for baseline information. Typically, in order to avoid maintaining a long list of 'tsids' for baseline information, it is RECOMMENDED that DOTS clients include in a request to update information related to a given target, the information of other targets (already communicated using a lower 'tsid' value) (assuming this fits within one single datagram). This update request will override these existing requests and hence optimize the number of 'tsid' request per DOTS client.

If no target clause in included in the request, this is an indication that the baseline information applies for the DOTS client domain as a whole.

An example of a PUT request to convey the baseline information is shown in Figure 19.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=126"
Content-Format: "application/dots+cbor"
```


```
{
  "ietf-dots-telemetry:telemetry": {
    "baseline": {
      "id": 1,
      "target-prefix": [
        "2001:db8:6401::1/128",
        "2001:db8:6401::2/128"
      ],
      "total-traffic-normal-baseline": {
        "unit": "megabytes-ps",
        "protocol": 6,
        "peak-g": "50"
      }
    }
  }
}
```


Figure 19: PUT to Convey the DOTS Traffic Baseline

6.3.2. Retrieve Installed Normal Traffic Baseline

A GET request with 'tsid' Uri-Path parameter is used to retrieve a specific installed DOTS client domain baseline traffic information. The same procedure as defined in (Section 6.1.3) is followed.

To retrieve all baseline information bound to a DOTS client, the DOTS client proceeds as specified in Section 6.1.1.

6.3.3. Delete Installed Normal Traffic Baseline

A DELETE request is used to delete the installed DOTS client domain normal traffic baseline. The same procedure as defined in (Section 6.1.4) is followed.

6.4. Reset Installed Telemetry Setup

Upon bootstrapping (or reboot or any other event that may alter the DOTS client setup), a DOTS client MAY send a DELETE request to set the telemetry parameters to default values. Such a request does not

include any 'tsid'. An example of such request is depicted in Figure 20.

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

Figure 20: Delete Telemetry Configuration

6.5. Conflict with Other DOTS Clients of the Same Domain

A DOTS server may detect conflicts between requests to convey pipe and baseline information received from DOTS clients of the same DOTS client domain. 'conflict-information' is used to report the conflict to the DOTS client following similar conflict handling discussed in Section 4.4.1 of [I-D.ietf-dots-signal-channel]. The conflict cause can be set to one of these values:

1: Overlapping targets (already defined in [I-D.ietf-dots-signal-channel]).

TBA: Overlapping pipe scope (see Section 11).

7. DOTS Pre-or-Ongoing Mitigation Telemetry

There are two broad types of DDoS attacks, one is bandwidth consuming attack, the other is target resource consuming attack. This section outlines the set of DOTS telemetry attributes (Section 7.1) that covers both the types of attacks. The ultimate objective of these attributes is to allow for the complete knowledge of attacks and the various particulars that can best characterize attacks.

The "ietf-dots-telemetry" YANG module (Section 9) augments the "ietf-dots-signal" with a new message type called "telemetry". The tree structure of the "telemetry" message type is shown Figure 23.

The pre-or-ongoing-mitigation telemetry attributes are indicated by the path-suffix '/tm'. The '/tm' is appended to the path-prefix to form the URI used with a CoAP request to signal the DOTS telemetry. Pre-or-ongoing-mitigation telemetry attributes specified in Section 7.1 can be signaled between DOTS agents.

Pre-or-ongoing-mitigation telemetry attributes may be sent by a DOTS client or a DOTS server.

DOTS agents SHOULD bind pre-or-ongoing-mitigation telemetry data with mitigation requests relying upon the target clause. In particular, a telemetry PUT request sent after a mitigation request may include a reference to that mitigation request ('mid-list') as shown in Figure 21. An example illustrating requests correlation by means of 'target-prefix' is shown in Figure 22.

When generating telemetry data to send to a peer, the DOTS agent must auto-scale so that appropriate unit(s) are used.

```
+-----------+                                       +-----------+
|DOTS client|                                       |DOTS server|
+-----------+                                       +-----------+
      |                                                   |
      |=========Mitigation Request (mid)====================>|
      |                                                   |
      |================ Telemetry (mid-list{mid})============>|
      |                                                   |
```

Figure 21: Example of Request Correlation using 'mid'

```
+-----------+                                       +-----------+
|DOTS client|                                       |DOTS server|
+-----------+                                       +-----------+
      |                                                   |
      |<=============== Telemetry (target-prefix)=============|
      |                                                   |
      |=========Mitigation Request (target-prefix)===========>|
      |                                                   |
```

Figure 22: Example of Request Correlation using Target Prefix

DOTS agents MUST NOT sent pre-or-ongoing-mitigation telemetry messages to the same peer more frequently than once every 'telemetry-notify-interval' (Section 6.1).

DOTS pre-or-ongoing-mitigation telemetry request and response messages MUST be marked as Non-Confirmable messages.

```
augment /ietf-signal:dots-signal/ietf-signal:message-type:
  +--:(telemetry-setup) {dots-telemetry}?
  |  +--rw telemetry* [cuid tsid]
  |  ...
  +--:(telemetry) {dots-telemetry}?
     +--rw pre-or-ongoing-mitigation* [cuid tmid]
        +--rw cuid                       string
        +--rw cdid?                      string
        +--rw tmid               uint32
        +--rw target
        |  ...
        +--rw total-traffic* [unit protocol]
        |  ...
        +--rw total-attack-traffic* [unit protocol]
        |  ...
        +--rw total-attack-connection
        |  ...
        +--rw attack-detail
           ...
```

Figure 23: Telemetry Message Type Tree Structure

7.1. Pre-or-Ongoing-Mitigation DOTS Telemetry Attributes

The description and motivation behind each attribute are presented in Section 3. DOTS telemetry attributes are optionally signaled and therefore MUST NOT be treated as mandatory fields in the DOTS signal channel protocol.

7.1.1. Target

A target resource (Figure 24) is identified using the attributes 'target-prefix', 'target-port-range', 'target-protocol', 'target-fqdn', 'target-uri', or 'alias-name' defined in the base DOTS signal channel protocol.

```
+--:(telemetry) {dots-telemetry}?
   +--rw pre-or-ongoing-mitigation* [cuid tmid]
      +--rw cuid                       string
      +--rw cdid?                      string
      +--rw tmid               uint32
      +--rw target
      |  +--rw target-prefix*       inet:ip-prefix
      |  +--rw target-port-range* [lower-port]
      |  |  +--rw lower-port    inet:port-number
      |  |  +--rw upper-port?   inet:port-number
      |  +--rw target-protocol*     uint8
      |  +--rw target-fqdn*         inet:domain-name
      |  +--rw target-uri*          inet:uri
      |  +--rw alias-name*          string
      |  +--rw mid-list*            uint32
      +--rw total-traffic* [unit protocol]
      |  ...
      +--rw total-attack-traffic* [unit protocol]
      |  ...
      +--rw total-attack-connection
      |  ...
      +--rw attack-detail
         ...
```

Figure 24: Target Tree Structure

At least one of the attributes 'target-prefix', 'target-fqdn', 'target-uri', 'alias-name', or 'mid-list' MUST be present in the target definition.

If the target is subjected to bandwidth consuming attack, the attributes representing the percentile values of the 'attack-id' attack traffic are included.

If the target is subjected to resource consuming DDoS attacks, the same attributes defined for Section 7.1.4 are applicable for representing the attack.

This is an optional sub-attribute.

7.1.2. Total Traffic

This attribute (Figure 25) conveys the percentile values of total traffic observed during a DDoS attack.

The total traffic is represented for a target and is transport-protocol specific.

```
       +--:(telemetry) {dots-telemetry}?
          +--rw pre-or-ongoing-mitigation* [cuid tmid]
             +--rw cuid                       string
             +--rw cdid?                      string
             +--rw tmid               uint32
             +--rw target
             |  ...
             +--rw total-traffic* [unit protocol]
             |  +--rw unit                 unit
             |  +--rw protocol             uint8
             |  +--rw low-percentile-g?    yang:gauge64
             |  +--rw mid-percentile-g?    yang:gauge64
             |  +--rw high-percentile-g?   yang:gauge64
             |  +--rw peak-g?              yang:gauge64
             +--rw total-attack-traffic* [unit protocol]
             |  ...
             +--rw total-attack-connection
             |  ...
             +--rw attack-detail
                ...
```

Figure 25: Total Traffic Tree Structure

7.1.3. Total Attack Traffic

This attribute (Figure 26) conveys the total attack traffic identified by the DOTS client domain's DMS (or DDoS Detector).

The total attack traffic is represented for a target and is transport-protocol specific.

```
+--:(telemetry) {dots-telemetry}?
   +--rw pre-or-ongoing-mitigation* [cuid tmid]
      +--rw cuid                       string
      +--rw cdid?                      string
      +--rw tmid               uint32
      +--rw target
      |  ...
      +--rw total-traffic* [unit protocol]
      |  ...
      +--rw total-attack-traffic* [unit protocol]
      |  +--rw unit                 unit
      |  +--rw protocol             uint8
      |  +--rw low-percentile-g?    yang:gauge64
      |  +--rw mid-percentile-g?    yang:gauge64
      |  +--rw high-percentile-g?   yang:gauge64
      |  +--rw peak-g?              yang:gauge64
      +--rw total-attack-connection
      |  ...
      +--rw attack-detail
         ...
```

Figure 26: Total Attack Traffic Tree Structure

7.1.4. Total Attack Connections

If the target is subjected to resource consuming DDoS attack, this attribute is used to convey the percentile values of total attack connections. The following optional sub-attributes for the target per transport-protocol are included to represent the attack characteristics:

- o The number of simultaneous attack connections to the target.
- o The number of simultaneous embryonic connections to the target.
- o The number of attack connections per second to the target.
- o The number of attack requests to the target.

```
+--:(telemetry) {dots-telemetry}?
   +--rw pre-or-ongoing-mitigation* [cuid tmid]
      +--rw cuid                       string
      +--rw cdid?                      string
      +--rw tmid               uint32
      +--rw target
      |  ...
      +--rw total-traffic* [unit protocol]
      |  ...
      +--rw total-attack-traffic* [unit protocol]
      |  ...
      +--rw total-attack-connection
      |  +--rw low-percentile-l* [protocol]
      |  |  +--rw protocol              uint8
      |  |  +--rw connection?           yang:gauge64
      |  |  +--rw embryonic?            yang:gauge64
      |  |  +--rw connection-ps?        yang:gauge64
      |  |  +--rw request-ps?           yang:gauge64
      |  |  +--rw partial-request-ps?   yang:gauge64
      |  +--rw mid-percentile-l* [protocol]
      |  |  +--rw protocol              uint8
      |  |  +--rw connection?           yang:gauge64
      |  |  +--rw embryonic?            yang:gauge64
      |  |  +--rw connection-ps?        yang:gauge64
      |  |  +--rw request-ps?           yang:gauge64
      |  |  +--rw partial-request-ps?   yang:gauge64
      |  +--rw high-percentile-l* [protocol]
      |  |  +--rw protocol              uint8
      |  |  +--rw connection?           yang:gauge64
      |  |  +--rw embryonic?            yang:gauge64
      |  |  +--rw connection-ps?        yang:gauge64
      |  |  +--rw request-ps?           yang:gauge64
      |  |  +--rw partial-request-ps?   yang:gauge64
      |  +--rw peak-l* [protocol]
      |     +--rw protocol              uint8
      |     +--rw connection?           yang:gauge64
      |     +--rw embryonic?            yang:gauge64
      |     +--rw connection-ps?        yang:gauge64
      |     +--rw request-ps?           yang:gauge64
      |     +--rw partial-request-ps?   yang:gauge64
      +--rw attack-detail
         ...
```

Figure 27: Total Attack Connections Tree Structure

7.1.5. Attack Details

This attribute (Figure 28) is used to signal a set of details characterizing an attack. The following optional sub-attributes describing the on-going attack can be signal as attack details.

id: Vendor ID is a security vendor's Enterprise Number as registered with IANA [Enterprise-Numbers]. It is a four-byte integer value.

attack-id: Unique identifier assigned by the vendor for the attack.

attack-name: 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. Textual representation of attack solves two problems: (a) avoids the need to create mapping tables manually between vendors and (2) avoids the need to standardize attack types which keep evolving.

attack-severity: Attack severity. These values are supported: Emergency (1), critical (2), and alert (3).

start-time: The time the attack started. The attack's start time is expressed in seconds relative to 1970-01-01T00:00Z in UTC time (Section 2.4.1 of [RFC7049]). The CBOR encoding is modified so that the leading tag 1 (epoch-based date/time) MUST be omitted.

end-time: The time the attack-id attack ended. The attack end time is expressed in seconds relative to 1970-01-01T00:00Z in UTC time (Section 2.4.1 of [RFC7049]). The CBOR encoding is modified so that the leading tag 1 (epoch-based date/time) MUST be omitted.

source-count: A count of sources involved in the attack targeting the victim.

top-talkers: A list of top talkers among attack sources. The top talkers are represented using the 'source-prefix'.

'spoofed-status' is used whether a top talker is a spoofed IP address (e.g., reflection attacks) or not.

If the target is subjected to bandwidth consuming attack, the attack traffic from each of the top talkers is included ('total-attack-traffic', Section 7.1.3).

If the target is subjected to resource consuming DDoS attacks, the same attributes defined for Section 7.1.4 are applicable for representing the attack per talker.

```
       +--:(telemetry) {dots-telemetry}?
          +--rw pre-or-ongoing-mitigation* [cuid tmid]
             +--rw cuid                       string
             +--rw cdid?                      string
             +--rw tmid               uint32
             +--rw target
             |  ...
             +--rw total-traffic* [unit protocol]
             |  ...
             +--rw total-attack-traffic* [unit protocol]
             |  ...
             +--rw total-attack-connection
             |  ...
             +--rw attack-detail
                +--rw id?                uint32
                +--rw attack-id?         string
                +--rw attack-name?       string
                +--rw attack-severity?   attack-severity
                +--rw start-time?        uint64
                +--rw end-time?          uint64
                +--rw source-count
                |  +--rw low-percentile-g?    yang:gauge64
                |  +--rw mid-percentile-g?    yang:gauge64
                |  +--rw high-percentile-g?   yang:gauge64
                |  +--rw peak-g?              yang:gauge64
                +--rw top-talker
                   +--rw talker* [source-prefix]
                      +--rw spoofed-status?            boolean
                      +--rw source-prefix              inet:ip-prefix
                      +--rw source-port-range* [lower-port]
                      |  +--rw lower-port    inet:port-number
                      |  +--rw upper-port?   inet:port-number
                      +--rw source-icmp-type-range*    [lower-type]
                      |  +--rw lower-type    uint8
                      |  +--rw upper-type?   uint8
                      +--rw total-attack-traffic* [unit]
                      |  +--rw unit                  unit
                      |  +--rw low-percentile-g?    yang:gauge64
                      |  +--rw mid-percentile-g?    yang:gauge64
                      |  +--rw high-percentile-g?   yang:gauge64
                      |  +--rw peak-g?              yang:gauge64
                      +--rw total-attack-connection
                         +--rw low-percentile-l* [protocol]
                         |  ...
                         +--rw mid-percentile-l* [protocol]
                         |  ...
                         +--rw high-percentile-l* [protocol]
                         |  ...
```

```
                         +--rw peak-l* [protocol]
                            ...
```

Figure 28: Attack Detail Tree Structure

7.2. From DOTS Clients to DOTS Servers

DOTS clients uses PUT request to signal pre-or-ongoing-mitigation telemetry to DOTS servers. An example of such request is shown in Figure 29.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=123"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry": {
    "pre-or-ongoing-mitigation": {
      "target": {
        {
          "target-prefix": [
          "2001:db8::1/128"
          ]
          "total-attack-traffic": [
            {
              "protocol": 17,
              "unit": "megabytes-ps",
              "mid-percentile-g": "900"
            }
          ],
          "attack-detail": {
            "start-time": "1957811234",
            "attack-severity": "emergency"
          }
        }
     }
    }
  }
}
```

Figure 29: PUT to Send Pre-or-Ongoing-Mitigation Telemetry

'cuid' is a mandatory Uri-Path parameter for PUT requests.

The following additional Uri-Path parameter is defined:

tmid: Telemetry Identifier is an identifier for the DOTS pre-or-ongoing-mitigation telemetry data represented as an integer. This identifier MUST be generated by DOTS clients. 'tsid' values MUST increase monotonically (when a new PUT is generated by a DOTS client to convey pre-or-ongoing-mitigation telemetry).

This is a mandatory attribute.

At least 'target' attribute and another pre-or-ongoing-mitigation attributes (Section 7.1) MUST be present in the PUT request. If only the 'target' attribute is present, this request is handled as per Section 7.3.

The relative order of two PUT requests carrying DOTS pre-or-ongoing-mitigation telemetry from a DOTS client is determined by comparing their respective 'tmid' values. If such two requests have overlapping 'target', the PUT request with higher numeric 'tmid' value will override the request with a lower numeric 'tmid' value. The overlapped lower numeric 'tmid' MUST be automatically deleted and no longer be available.

The DOTS server indicates the result of processing a PUT request using CoAP response codes. The response code 2.04 (Changed) is returned if the DOTS server has accepted the pre-or-ongoing-mitigation telemetry. The error response code 5.03 (Service Unavailable) is returned if the DOTS server has erred. 5.03 uses Max-Age option to indicate the number of seconds after which to retry.

How long a DOTS server maintains a 'tmid' as active or logs the enclosed telemetry information is implementation-specific. Note that if a 'tmid' is still active, then logging details are updated by the DOTS server as a function of the updates received from the peer DOTS client.

A DOTS client that lost the state of its active 'tmids' or has to set 'tmid' back to zero (e.g., crash or restart) MUST send a GET request to the DOTS server to retrieve the list of active 'tmid'. The DOTS client may then delete 'tmids' that should not be active anymore.

7.3. From DOTS Servers to DOTS Clients

The pre-or-ongoing-mitigation (attack details, in particular) can also be signaled from DOTS servers to DOTS clients. For example, the DOTS server co-located with a DDoS detector collects monitoring information from the target network, identifies DDoS attack using

statistical analysis or deep learning techniques, and signals the attack details to the DOTS client.

The DOTS client can use the attack details to decide whether to trigger a DOTS mitigation request or not. Furthermore, the security operation personnel at the DOTS client domain can use the attack details to determine the protection strategy and select the appropriate DOTS server for mitigating the attack.

In order to receive pre-or-ongoing-mitigation telemetry notifications from a DOTS server, a DOTS client MUST send a PUT (followed by a GET) with the target filter. An example of such PUT request is shown in Figure 30. In order to avoid maintaining a long list of such requests, it is RECOMMENDED that DOTS clients include all targets in the same request. DOTS servers may be instructed to restrict the number of pre-or-ongoing-mitigation requests per DOTS client domain.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=123"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry": {
    "pre-or-ongoing-mitigation": {
      "target": {
        {
          "target-prefix": [
            "2001:db8::/32"
           ]
        }
      }
    }
  }
}
```

Figure 30: PUT to Request Pre-or-Ongoing-Mitigation Telemetry

DOTS clients of the same domain can request to receive pre-or-ongoing-mitigation telemetry bound to the same target.

The DOTS client conveys the Observe Option set to '0' in the GET request to receive asynchronous notifications carrying pre-or-ongoing-mitigation telemetry data from the DOTS server. The GET request specify a 'tmid' (Figure 31) or not (Figure 32).

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=123"
Observe: 0
```

Figure 31: GET to Subscribe to Telemetry Asynchronous Notifications for a Specific 'tmid'

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Observe: 0
```

Figure 32: GET to Subscribe to Telemetry Asynchronous Notifications for All 'tmids'

The DOTS server will send asynchronous notifications to the DOTS client when an event if following similar considerations as in Section 4.4.2.1 of [I-D.ietf-dots-signal-channel]. An example of a pre-or-ongoing-mitigation telemetry notification is shown in Figure 33.

```
{
  "ietf-dots-telemetry:telemetry": {
   "pre-or-ongoing-mitigation": {
      "target": {
        {
          "tmid": 123,
          "target-prefix": [
            "2001:db8::1/128"
           ]
          "total-attack-traffic": [
            {
              "protocol": 17,
              "unit": "megabytes-ps",
              "mid-percentile-g": "900"
            }
          ],
          "attack-detail": {
            "start-time": "1957818434",
            "attack-severity": "emergency"
          }
        }
      }
     }
   }
 }
```


Figure 33: Message Body of a Pre-or-Ongoing-Mitigation Telemetry Notification from the DOTS Server

A DOTS server may aggregate pre-or-ongoing-mitigation data (e.g., 'top-talkers') for all targets of a domain, or when justified, send specific information (e.g., 'top-talkers') per individual targets.

The DOTS client may log pre-or-ongoing-mitigation telemetry data with an alert sent to an administrator or a network controller. The DOTS client may send a mitigation request if the attack cannot be handled locally.

8. DOTS Telemetry Mitigation Status Update

8.1. DOTS Clients to Servers Mitigation Efficacy DOTS Telemetry Attributes

The mitigation efficacy telemetry attributes can be signaled from DOTS clients to DOTS servers as part of the periodic mitigation efficacy updates to the server (Section 5.3.4 of [I-D.ietf-dots-signal-channel]).

Total Attack Traffic: The overall attack traffic as observed from the DOTS client perspective during an active mitigation. See Figure 26.

Attack Details: The overall attack details as observed from the DOTS client perspective during an active mitigation. See Section 7.1.5.

The "ietf-dots-telemetry" YANG module augments the "mitigation-scope" type message defined in "ietf-dots-signal" so that these attributes can be signalled by a DOTS client in a mitigation efficacy update (Figure 34).

```
  augment /ietf-signal:dots-signal/ietf-signal:message-type
          /ietf-signal:mitigation-scope/ietf-signal:scope:
    +--rw total-attack-traffic* [unit] {dots-telemetry}?
    |  ...
    +--rw attack-detail {dots-telemetry}?
       +--rw id?                uint32
       +--rw attack-id?         string
       +--rw attack-name?       string
       +--rw attack-severity?   attack-severity
       +--rw start-time?        uint64
       +--rw end-time?          uint64
       +--rw source-count
       |  ...
       +--rw top-talker
          ...
```

Figure 34: Telemetry Efficacy Update Tree Structure

In order to signal telemetry data in a mitigation efficacy update, it is RECOMMENDED that the DOTS client has already established a DOTS telemetry setup session with the server in 'idle' time.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=123"
If-Match:
Content-Format: "application/dots+cbor"
```


```
{
 "ietf-dots-signal-channel:mitigation-scope": {
   "scope": [
     {
       "alias-name": [
          "myserver"
        ],
       "attack-status": "under-attack",
       "ietf-dots-telemetry:total-attack-traffic": [
         {
           "ietf-dots-telemetry:unit": "megabytes-ps",
           "ietf-dots-telemetry:mid-percentile-g": "900"
         }
       ]
     }
   ]
 }
}
```


Figure 35: An Example of Mitigation Efficacy Update with Telemetry Attributes

8.2. DOTS Servers to Clients Mitigation Status DOTS Telemetry Attributes

The mitigation status telemetry attributes can be signaled from the DOTS server to the DOTS client as part of the periodic mitigation status update (Section 5.3.3 of [I-D.ietf-dots-signal-channel]). In particular, DOTS clients can receive asynchronous notifications of the attack details from DOTS servers using the Observe option defined in [RFC7641].

In order to make use of this feature, DOTS clients MUST establish a telemetry setup session with the DOTS server in 'idle' time and MUST set the 'server-originated-telemetry' attribute to 'true'.

DOTS servers MUST NOT include telemetry attributes in mitigation status updates sent to DOTS clients for which 'server-originated-telemetry' attribute is set to 'false'.

As defined in [RFC8612], the actual mitigation activities can include several countermeasure mechanisms. The DOTS server signals the current operational status to each relevant countermeasure. A list of attacks detected by each countermeasure MAY also be included. The same attributes defined for Section 7.1.5 are applicable for describing the attacks detected and mitigated.

The "ietf-dots-telemetry" YANG module (Section 9) augments the "mitigation-scope" type message defined in "ietf-dots-signal" with telemetry data as depicted in following tree structure:

```
  augment /ietf-signal:dots-signal/ietf-signal:message-type
          /ietf-signal:mitigation-scope/ietf-signal:scope:
    +--ro total-traffic* [unit protocol] {dots-telemetry}?
    |  +--ro unit                 unit
    |  +--ro protocol             uint8
    |  +--ro low-percentile-g?    yang:gauge64
    |  +--ro mid-percentile-g?    yang:gauge64
    |  +--ro high-percentile-g?   yang:gauge64
    |  +--ro peak-g?              yang:gauge64
    +--rw total-attack-traffic* [unit] {dots-telemetry}?
    |  +--rw unit                 unit
    |  +--rw low-percentile-g?    yang:gauge64
    |  +--rw mid-percentile-g?    yang:gauge64
    |  +--rw high-percentile-g?   yang:gauge64
    |  +--rw peak-g?              yang:gauge64
    +--ro total-attack-connection {dots-telemetry}?
    |  +--ro low-percentile-c
    |  |  +--ro connection?           yang:gauge64
    |  |  +--ro embryonic?            yang:gauge64
    |  |  +--ro connection-ps?        yang:gauge64
    |  |  +--ro request-ps?           yang:gauge64
    |  |  +--ro partial-request-ps?   yang:gauge64
    |  +--ro mid-percentile-c
    |  |  ...
    |  +--ro high-percentile-c
    |  |  ...
    |  +--ro peak-c
    |     ...
    +--rw attack-detail {dots-telemetry}?
       +--rw id?                 uint32
       +--rw attack-id?          string
       +--rw attack-name?        string
       +--rw attack-severity?   attack-severity
       +--rw start-time?         uint64
       +--rw end-time?           uint64
       +--rw source-count
       |  +--rw low-percentile-g?    yang:gauge64
```

```
          |  +--rw mid-percentile-g?    yang:gauge64
          |  +--rw high-percentile-g?   yang:gauge64
          |  +--rw peak-g?              yang:gauge64
          +--rw top-talker
             +--rw talker* [source-prefix]
                +--rw spoofed-status?           boolean
                +--rw source-prefix             inet:ip-prefix
                +--rw source-port-range* [lower-port]
                |  +--rw lower-port    inet:port-number
                |  +--rw upper-port?   inet:port-number
                +--rw source-icmp-type-range*    [lower-type]
                |  +--rw lower-type    uint8
                |  +--rw upper-type?   uint8
                +--rw total-attack-traffic* [unit]
                |  +--rw unit                 unit
                |  +--rw low-percentile-g?    yang:gauge64
                |  +--rw mid-percentile-g?    yang:gauge64
                |  +--rw high-percentile-g?   yang:gauge64
                |  +--rw peak-g?              yang:gauge64
                +--rw total-attack-connection
                   +--rw low-percentile-c
                   |  +--rw connection?           yang:gauge64
                   |  +--rw embryonic?            yang:gauge64
                   |  +--rw connection-ps?        yang:gauge64
                   |  +--rw request-ps?           yang:gauge64
                   |  +--rw partial-request-ps?   yang:gauge64
                   +--rw mid-percentile-c
                   |  ...
                   +--rw high-percentile-c
                   |  ...
                   +--rw peak-c
                      ...
```

Figure 36 shows an example of an asynchronous notification of attack mitigation status from the DOTS server. This notification signals both the mid-percentile value of processed attack traffic and the peak percentile value of unique sources involved in the attack.

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 12332,
        "mitigation-start": "1507818434",
        "alias-name": [
             "myserver"
        ],
        "lifetime": 1600,
        "status": "attack-successfully-mitigated",
        "bytes-dropped": "134334555",
        "bps-dropped": "43344",
        "pkts-dropped": "333334444",
        "pps-dropped": "432432",
        "ietf-dots-telemetry:total-attack-traffic": [
          {
            "ietf-dots-telemetry:unit": "megabytes-ps",
            "ietf-dots-telemetry:mid-percentile-g": "900"
          }
        ],
        "ietf-dots-telemetry::attack-detail": {
          "ietf-dots-telemetry:source-count": {
           "ietf-dots-telemetry:peak-g": "10000"
          }
        }
      }
    ]
  }
}
```


Figure 36: Response Body of a Mitigation Status With Telemetry Attributes

9. YANG Module

This module uses types defined in [RFC6991] and [RFC8345].

```
<CODE BEGINS> file "ietf-dots-telemetry@2020-03-08.yang"
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 SSSS: Distributed Denial-of-Service Open Threat
```

```
                 Signaling (DOTS) Signal Channel Specification";
  }
  import ietf-dots-data-channel {
    prefix ietf-data;
    reference
      "RFC DDDD: 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";
  }

  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 2020-03-08 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Telemetry";
  }

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

  typedef attack-severity {
    type enumeration {
      enum emergency {
        value 1;
        description
          "The attack is severe: emergency.";
      }
      enum critical {
        value 2;
        description
          "The attack is critical.";
      }
      enum alert {
        value 3;
        description
          "This is an alert.";
      }
    }
    description
      "Enumeration for attack severity.";
  }

  typedef unit {
    type enumeration {
      enum pps {
        value 1;
        description
```

```
          "Packets per second (PPS).";
      }
      enum kilo-pps {
        value 2;
        description
          "Kilo packets per second (Kpps).";
      }
      enum bps {
        value 3;
        description
          "Bit per Second (BPS).";
      }
      enum kilobyte-ps {
        value 4;
        description
          "Kilobyte per second.";
      }
      enum megabit-ps {
        value 5;
        description
          "Megabit per second.";
      }
      enum megabyte-ps {
        value 6;
        description
          "Megabyte per second.";
      }
      enum gigabit-ps {
        value 7;
        description
          "Gigabit per second.";
      }
      enum gigabyte-ps {
        value 8;
        description
          "Gigabyte per second.";
      }
      enum terabit-ps {
        value 9;
        description
          "Terabit per second.";
      }
      enum terabyte-ps {
        value 10;
        description
          "Terabyte per second.";
      }
    }
```

```
    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
          "Second.";
      }
      enum 5-seconds {
        value 2;
        description
          "5 seconds.";
      }
      enum 30-seconds {
        value 3;
        description
          "30 seconds.";
      }
```

```
      enum minute {
        value 4;
        description
          "One minute.";
      }
      enum 5-minutes {
        value 5;
        description
          "5 minutes.";
      }
      enum 10-minutes {
        value 6;
        description
          "10 minutes.";
      }
      enum 30-minutes {
        value 7;
        description
          "30 minutes.";
      }
      enum hour {
        value 8;
        description
          "One hour.";
      }
    }
    description
      "Enumeration to indicate the measurement perdiod.";
  }

  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.";
  }

  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 traffic.";
    }
    leaf mid-percentile-g {
      type yang:gauge64;
```

```
      description
        "Mid percentile.";
    }
    leaf high-percentile-g {
      type yang:gauge64;
      description
        "High percentile.";
    }
    leaf peak-g {
      type yang:gauge64;
      description
        "Peak";
    }
  }

  grouping unit-config {
    description
      "Generic grouping for unit configuration.";
    list unit-config {
      key "unit";
      description
        "Controls which units are allowed when sharing telemetry
         data.";
      leaf unit {
        type unit;
        description
          "The traffic can be measured in packets per
           second (PPS) or kilo packets per second (Kpps) and Bits per
           Second (BPS), and kilobytes per second or megabytes per second
           or gigabytes per second.";
      }
      leaf unit-status {
        type boolean;
        description
          "Enable/disable the use of the measurement unit.";
      }
    }
  }

  grouping traffic-unit {
    description
      "Grouping of traffic as a function of measurement unit.";
    leaf unit {
      type unit;
      description
        "The traffic can be measured in packets per
         second (PPS) or kilo packets per second (Kpps) and Bits per
         Second (BPS), and kilobytes per second or megabytes per second
```

```
         or gigabytes per second.";
    }
    uses percentile;
  }

  grouping traffic-unit-protocol {
    description
      "Grouping of traffic of a given transport protocol as
       a function of measurement unit.";
    leaf unit {
      type unit;
      description
        "The traffic can be measured in packets per
         second (PPS) or kilo packets per second (Kpps) and Bits per
         Second (BPS), and kilobytes per second or megabytes per second
         or gigabytes per second.";
    }
    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 field contains 6 for TCP,
         17 for UDP, 33 for DCCP, or 132 for SCTP.";
    }
    uses percentile;
  }

  grouping total-connection-capacity {
    description
      "Total Connections Capacity. If the target is subjected
       to resource consuming DDoS attack, 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. The threshold is
         transport-protocol specific because the target server
         could support multiple protocols.";
    }
    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 and 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 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-percentile {
    description
      "Total attack connections.";
    list low-percentile-l {
      key "protocol";
      description
        "Low percentile of 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;
    }
    list mid-percentile-l {
      key "protocol";
      description
        "Mid percentile of 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;
    }
```

```
    list high-percentile-l {
      key "protocol";
      description
        "Highg percentile of 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;
    }
    list peak-l {
      key "protocol";
      description
        "Peak 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 attack-detail {
    description
      "Various information and details that describe the on-going
       attacks that needs 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 id {
      type uint32;
      description
        "Vendor ID is a security vendor's Enterprise Number.";
    }
    leaf attack-id {
      type string;
      description
        "Unique identifier assigned by the vendor for the attack.";
    }
    leaf attack-name {
      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";
    }
    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-baseline {
      key "unit protocol";
      description
        "Total traffic normal baselines.";
```

```
      uses traffic-unit-protocol;
    }
    list total-connection-capacity {
      key "protocol";
      description
        "Total connection capacity.";
      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 pre-or-ongoing-mitigation {
    description
      "Grouping for the telemetry data.";
    list total-traffic {
      key "unit protocol";
      description
        "Total traffic.";
      uses traffic-unit-protocol;
    }
    list total-attack-traffic {
      key "unit protocol";
      description
        "Total attack traffic per protocol.";
      uses traffic-unit-protocol;
    }
    container total-attack-connection {
      description
        "Total attack connections.";
      uses connection-protocol-percentile;
    }
    container attack-detail {
      description
        "Attack details.";
      uses attack-detail;
      container top-talker {
        description
          "Top attack sources.";
        uses top-talker;
      }
    }
  }
```

```
  augment "/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.";
    list total-traffic {
      key "unit protocol";
      config false;
      description
        "Total traffic.";
      uses traffic-unit-protocol;
    }
    list total-attack-traffic {
      key "unit";
      description
        "Total attack traffic.";
      uses traffic-unit;
    }
    container total-attack-connection {
      config false;
      description
        "Total attack connections.";
      uses connection-percentile;
    }
    container attack-detail {
      description
        "Atatck details";
      uses attack-detail;
      container top-talker {
        description
          "Top attack sources.";
        uses top-talker-aggregate;
      }
    }
  }

  augment "/ietf-signal:dots-signal/ietf-signal:message-type" {
    if-feature "dots-telemetry";
    description
      "Add a new choice to enclose telemetry data in DOTS
       signal channel.";
    case telemetry-setup {
      description
        "Indicates the message is about telemetry.";
      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.";
            }
          }
          container max-config-values {
            config false;
            description
              "Maximum acceptable configuration values.";
            uses percentile-config;
            // Check if this is right place for indciating this capability
            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";
              }
              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;
            }
          }
          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 in packets per
                   second (PPS) or kilo packets per second (Kpps) and Bits per
                   Second (BPS), and kilobytes per second or megabytes per
second
                   or gigabytes per second.";
              }
            }
          }
          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 send 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;
     }
   }
 }
}
<CODE ENDS>
```

10. YANG/JSON Mapping Parameters to CBOR

All DOTS telemetry parameters in the payload of the DOTS signal channel MUST be mapped to CBOR types as shown in the following table:

- o Implementers may use the values in: https://github.com/boucadair/draft-dots-telemetry/blob/master/mapping-table.txt

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
tsid	uint32	TBA1	0 unsigned	Number
telemetry-config	container	TBA2	5 map	Object
low-percentile	decimal64	TBA3	6 tag 4 [-2, integer]	String
mid-percentile	decimal64	TBA4	6 tag 4 [-2, integer]	String
high-percentile	decimal64	TBA5	6 tag 4 [-2, integer]	String
unit-config	list	TBA6	4 array	Array
unit	enumeration	TBA7	0 unsigned	String
unit-status	boolean	TBA8	7 bits 20 7 bits 21	False True
total-pipe-capability	list	TBA9	4 array	Array
pipe	uint64	TBA10	0 unsigned	String
pre-or-ongoing-mitigation	list	TBA11	4 array	Array
total-traffic-normal-baseline	list	TBA12	4 array	Array
low-percentile-g	yang:gauge64	TBA13	0 unsigned	String
mid-percentile-g	yang:gauge64	TBA14	0 unsigned	String

high-percentile-g	yang:gauge64	TBA15	0 unsigned	String
peak-g	yang:gauge64	TBA16	0 unsigned	String
total-attack-traffic	list	TBA17	4 array	Array
total-traffic	list	TBA18	4 array	Array
total-connection-capacity	list	TBA19	4 array	Array
connection	uint64	TBA20	0 unsigned	String
connection-client	uint64	TBA21	0 unsigned	String
embryonic	uint64	TBA22	0 unsigned	String
embryonic-client	uint64	TBA23	0 unsigned	String
connection-ps	uint64	TBA24	0 unsigned	String
connection-client-ps	uint64	TBA25	0 unsigned	String
request-ps	uint64	TBA26	0 unsigned	String
request-client-ps	uint64	TBA27	0 unsigned	String
partial-request-ps	uint64	TBA28	0 unsigned	String
partial-request-client-ps	uint64	TBA29	0 unsigned	String
total-attack-connection	container	TBA30	5 map	Object
low-percentile-l	list	TBA31	4 array	Array
mid-percentile-l	list	TBA32	4 array	Array
high-percentile-l	list	TBA33	4 array	Array
peak-l	list	TBA34	4 array	Array
attack-detail	container	TBA35	5 map	Object
id	uint32	TBA36	0 unsigned	Number
attack-id	string	TBA37	3 text string	String
attack-name	string	TBA38	3 text string	String
attack-severity	enumeration	TBA39	0 unsigned	String
start-time	uint64	TBA40	0 unsigned	String
end-time	uint64	TBA41	0 unsigned	String
source-count	container	TBA42	5 map	Object
top-talker	container	TBA43	5 map	Object
spoofed-status	boolean	TBA44	7 bits 20 7 bits 21	False True
low-percentile-c	container	TBA45	5 map	Object
mid-percentile-c	container	TBA46	5 map	Object
high-percentile-c	container	TBA47	5 map	Object
peak-c	container	TBA48	5 map	Object
baseline	container	TBA49	5 map	Object
current-config	container	TBA50	5 map	Object
max-config-values	container	TBA51	5 map	Object
min-config-values	container	TBA52	5 map	Object
supported-units	container	TBA53	5 map	Object
server-originated-telemetry	boolean	TBA54	7 bits 20 7 bits 21	False True
telemetry-notify-interval	uint32	TBA55	0 unsigned	Number
tmid	uint32	TBA56	0 unsigned	Number

measurement-interval	identityref	TBA57	0 unsigned	String
measurement-sample	identityref	TBA58	0 unsigned	String
talker	list	TBA59	4 array	Array
source-prefix	inet: ip-prefix	TBA60	3 text string	String
mid-list	leaf-list	TBA61	4 array	Array
	uint32		0 unsigned	Number
source-port-range	list	TBA62	4 array	Array
source-icmp-type-range	list	TBA63	4 array	Array
lower-type	uint8	TBA64	0 unsigned	Number
upper-type	uint8	TBA65	0 unsigned	Number
target	container	TBA66	5 map	Object
ietf-dots-telemetry: telemetry	container	TBA67	5 map	Object
ietf-dots-telemetry: telemetry-setup	container	TBA68	5 map	Object
ietf-dots-telemetry: total-traffic	list	TBA69	4 array	Array
ietf-dots-telemetry: unit	enumeration	TBA70	0 unsigned	String
ietf-dots-telemetry: low-percentile-g	yang:gauge64	TBA71	0 unsigned	String
ietf-dots-telemetry: mid-percentile-g	yang:gauge64	TBA72	0 unsigned	String
ietf-dots-telemetry: high-percentile-g	yang:gauge64	TBA73	0 unsigned	String
ietf-dots-telemetry: peak-g	yang:gauge64	TBA74	0 unsigned	String
ietf-dots-telemetry: total-attack-traffic	list	TBA75	4 array	Array
ietf-dots-telemetry: total-attack-connection	container	TBA76	5 map	Object
ietf-dots-telemetry: low-percentile-c	container	TBA77	5 map	Object
ietf-dots-telemetry: mid-percentile-c	container	TBA78	5 map	Object
ietf-dots-telemetry: high-percentile-c	container	TBA79	5 map	Object
ietf-dots-telemetry: peak-c	container	TBA80	5 map	Object
ietf-dots-telemetry: connection	uint64	TBA81	0 unsigned	String
ietf-dots-telemetry: embryonic	uint64	TBA82	0 unsigned	String
ietf-dots-telemetry: connection-ps	uint64	TBA83	0 unsigned	String

ietf-dots-telemetry: request-ps	uint64	TBA84	0 unsigned	String
ietf-dots-telemetry: partial-request-ps	uint64	TBA85	0 unsigned	String
ietf-dots-telemetry: attack-detail	container	TBA86	5 map	Object
ietf-dots-telemetry: id	uint32	TBA87	0 unsigned	Number
ietf-dots-telemetry: attack-id	string	TBA88	3 text string	String
ietf-dots-telemetry: attack-name	string	TBA89	3 text string	String
ietf-dots-telemetry: attack-severity	enumeration	TBA90	0 unsigned	String
ietf-dots-telemetry: start-time	uint64	TBA91	0 unsigned	String
ietf-dots-telemetry: end-time	uint64	TBA92	0 unsigned	String
ietf-dots-telemetry: source-count	container	TBA93	5 map	Object
ietf-dots-telemetry: top-talker	container	TBA94	5 map	Object
ietf-dots-telemetry: spoofed-status	boolean	TBA95	7 bits 20 7 bits 21	False True
ietf-dots-telemetry: talker	list	TBA96	4 array	Array
ietf-dots-telemetry: source-prefix	inet: ip-prefix	TBA97	3 text string	String
ietf-dots-telemetry: source-port-range	list	TBA98	4 array	Array
ietf-dots-telemetry: lower-port	inet: port-number	TBA99	0 unsigned	Number
ietf-dots-telemetry: upper-port	inet: port-number	TBA100	0 unsigned	Number
ietf-dots-telemetry: source-icmp-type-range	list	TBA101	4 array	Array
ietf-dots-telemetry: lower-type	uint8	TBA102	0 unsigned	Number
ietf-dots-telemetry: upper-type	uint8	TBA103	0 unsigned	Number

11. IANA Considerationsr

11.1. DOTS Signal Channel CBOR Key Values

This specification registers the DOTS telemetry attributes in the IANA "DOTS Signal Channel CBOR Key Values" registry available at https://www.iana.org/assignments/dots/dots.xhtml#dots-signal-channel-cbor-key-values.

The DOTS telemetry attributes defined in this specification are comprehension-optional parameters.

- Note to the RFC Editor: (1) CBOR keys are assigned from the 32768-49151 range. (2) Please assign the following suggested values.

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document(s)
tsid	TBA1	0	IESG	[RFCXXXX]
telemetry-config	TBA2	5	IESG	[RFCXXXX]
low-percentile	TBA3	6tag4	IESG	[RFCXXXX]
mid-percentile	TBA4	6tag4	IESG	[RFCXXXX]
high-percentile	TBA5	6tag4	IESG	[RFCXXXX]
unit-config	TBA6	4	IESG	[RFCXXXX]
unit	TBA7	0	IESG	[RFCXXXX]
unit-status	TBA8	7	IESG	[RFCXXXX]
total-pipe-capability	TBA9	4	IESG	[RFCXXXX]
pipe	TBA10	0	IESG	[RFCXXXX]
pre-or-ongoing-mitigation	TBA11	4	IESG	[RFCXXXX]
total-traffic-normal-baseline	TBA12	4	IESG	[RFCXXXX]
low-percentile-g	TBA13	0	IESG	[RFCXXXX]
mid-percentile-g	TBA14	0	IESG	[RFCXXXX]
high-percentile-g	TBA15	0	IESG	[RFCXXXX]
peak-g	TBA16	0	IESG	[RFCXXXX]
total-attack-traffic	TBA17	4	IESG	[RFCXXXX]
total-traffic	TBA18	4	IESG	[RFCXXXX]
total-connection-capacity	TBA19	4	IESG	[RFCXXXX]
connection	TBA20	0	IESG	[RFCXXXX]
connection-client	TBA21	0	IESG	[RFCXXXX]
embryonic	TBA22	0	IESG	[RFCXXXX]
embryonic-client	TBA23	0	IESG	[RFCXXXX]
connection-ps	TBA24	0	IESG	[RFCXXXX]

connection-client-ps	TBA25	0	IESG	[RFCXXXX]
request-ps	TBA26	0	IESG	[RFCXXXX]
request-client-ps	TBA27	0	IESG	[RFCXXXX]
partial-request-ps	TBA28	0	IESG	[RFCXXXX]
partial-request-client-ps	TBA29	0	IESG	[RFCXXXX]
total-attack-connection	TBA30	5	IESG	[RFCXXXX]
low-percentile-l	TBA31	4	IESG	[RFCXXXX]
mid-percentile-l	TBA32	4	IESG	[RFCXXXX]
high-percentile-l	TBA33	4	IESG	[RFCXXXX]
peak-l	TBA34	4	IESG	[RFCXXXX]
attack-detail	TBA35	5	IESG	[RFCXXXX]
id	TBA36	0	IESG	[RFCXXXX]
attack-id	TBA37	3	IESG	[RFCXXXX]
attack-name	TBA38	3	IESG	[RFCXXXX]
attack-severity	TBA39	0	IESG	[RFCXXXX]
start-time	TBA40	0	IESG	[RFCXXXX]
end-time	TBA41	0	IESG	[RFCXXXX]
source-count	TBA42	5	IESG	[RFCXXXX]
top-talker	TBA43	5	IESG	[RFCXXXX]
spoofed-status	TBA44	7	IESG	[RFCXXXX]
low-percentile-c	TBA45	5	IESG	[RFCXXXX]
mid-percentile-c	TBA46	5	IESG	[RFCXXXX]
high-percentile-c	TBA47	5	IESG	[RFCXXXX]
peak-c	TBA48	5	IESG	[RFCXXXX]
ietf-dots-signal-cha	TBA49	5	IESG	[RFCXXXX]
current-config	TBA50	5	IESG	[RFCXXXX]
max-config-value	TBA51	5	IESG	[RFCXXXX]
min-config-values	TBA52	5	IESG	[RFCXXXX]
supported-units	TBA55	5	IESG	[RFCXXXX]
server-originated-telemetry	TBA54	7	IESG	[RFCXXXX]
telemetry-notify-interval	TBA55	0	IESG	[RFCXXXX]
tmid	TBA56	0	IESG	[RFCXXXX]
measurement-interval	TBA57	0	IESG	[RFCXXXX]
measurement-sample	TBA58	0	IESG	[RFCXXXX]
talker	TBA59	0	IESG	[RFCXXXX]
source-prefix	TBA60	0	IESG	[RFCXXXX]
mid-list	TBA61	4	IESG	[RFCXXXX]
source-port-range	TBA62	4	IESG	[RFCXXXX]
source-icmp-type-	TBA63	4	IESG	[RFCXXXX]
lower-type	TBA64	0	IESG	[RFCXXXX]
upper-type	TBA65	0	IESG	[RFCXXXX]
target	TBA66	5	IESG	[RFCXXXX]
ietf-dots-telemetry:telemetry	TBA67	5	IESG	[RFCXXXX]

ietf-dots-telemetry: telemetry-setup	TBA68	5	IESG	[RFCXXXX]
ietf-dots-telemetry: total-traffic	TBA69	0	IESG	[RFCXXXX]
ietf-dots-telemetry: unit	TBA70	0	IESG	[RFCXXXX]
ietf-dots-telemetry: low-percentile-g	TBA71	0	IESG	[RFCXXXX]
ietf-dots-telemetry: mid-percentile-g	TBA72	0	IESG	[RFCXXXX]
ietf-dots-telemetry: high-percentile-g	TBA73	0	IESG	[RFCXXXX]
ietf-dots-telemetry: peak-g	TBA74	0	IESG	[RFCXXXX]
ietf-dots-telemetry: total-attack-traffic	TBA75	0	IESG	[RFCXXXX]
ietf-dots-telemetry: total-attack-connection	TBA76	0	IESG	[RFCXXXX]
ietf-dots-telemetry: low-percentile-c	TBA77	0	IESG	[RFCXXXX]
ietf-dots-telemetry: mid-percentile-c	TBA78	0	IESG	[RFCXXXX]
ietf-dots-telemetry: high-percentile-c	TBA79	0	IESG	[RFCXXXX]
ietf-dots-telemetry: peak-c	TBA80	0	IESG	[RFCXXXX]
ietf-dots-telemetry: connection	TBA81	0	IESG	[RFCXXXX]
ietf-dots-telemetry: embryonic	TBA82	0	IESG	[RFCXXXX]
ietf-dots-telemetry: connection-ps	TBA83	0	IESG	[RFCXXXX]
ietf-dots-telemetry: request-ps	TBA84	0	IESG	[RFCXXXX]
ietf-dots-telemetry: partial-request-ps	TBA85	0	IESG	[RFCXXXX]
ietf-dots-telemetry: attack-detail	TBA86	0	IESG	[RFCXXXX]
ietf-dots-telemetry: id	TBA87	0	IESG	[RFCXXXX]
ietf-dots-telemetry: attack-id	TBA88	0	IESG	[RFCXXXX]
ietf-dots-telemetry: attack-name	TBA89	0	IESG	[RFCXXXX]
ietf-dots-telemetry: attack-severity	TBA90	0	IESG	[RFCXXXX]
ietf-dots-telemetry:	TBA91	0	IESG	[RFCXXXX]

start-time				
ietf-dots-telemetry: end-time	TBA92	0	IESG	[RFCXXXX]
ietf-dots-telemetry: source-count	TBA93	0	IESG	[RFCXXXX]
ietf-dots-telemetry: top-talker	TBA94	0	IESG	[RFCXXXX]
ietf-dots-telemetry: spoofed-status	TBA95	0	IESG	[RFCXXXX]
ietf-dots-telemetry: talker	TBA96	0	IESG	[RFCXXXX]
ietf-dots-telemetry: source-prefix	TBA97	0	IESG	[RFCXXXX]
ietf-dots-telemetry: source-port-range	TBA98	4	IESG	[RFCXXXX]
ietf-dots-telemetry: lower-port	TBA99	0	IESG	[RFCXXXX]
ietf-dots-telemetry: upper-port	TBA100	0	IESG	[RFCXXXX]
ietf-dots-telemetry: source-icmp-type-range	TBA101	4	IESG	[RFCXXXX]
ietf-dots-telemetry: lower-type	TBA102	0	IESG	[RFCXXXX]
ietf-dots-telemetry: upper-type	TBA103	0	IESG	[RFCXXXX]

11.2. DOTS Signal Channel Conflict Cause Codes

This specification requests IANA to assign a new code from the "DOTS Signal Channel Conflict Cause Codes" registry available at https://www.iana.org/assignments/dots/dots.xhtml#dots-signal-channel-conflict-cause-codes.

Code	Label	Description	Reference
TBA	overlapping-pipes	Overlapping pipe scope	[RFCXXXX]

11.3. DOTS Signal Telemetry YANG Module

This document requests IANA to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-dots-telemetry
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC7950] within the "YANG Parameters" registry.

```
name: ietf-dots-telemetry
namespace: urn:ietf:params:xml:ns:yang:ietf-dots-telemetry
maintained by IANA: N
prefix: dots-telemetry
reference: RFC XXXX
```

12. Security Considerations

Security considerations in [I-D.ietf-dots-signal-channel] need to be taken into consideration.

13. Contributors

The following individuals have contributed to this document:

o Li Su, CMCC, Email: suli@chinamobile.com

o Jin Peng, CMCC, Email: pengjin@chinamobile.com

o Pan Wei, Huawei, Email: william.panwei@huawei.com

14. Acknowledgements

The authors would like to thank Flemming Andreasen, Liang Xia, and Kaname Nishizuka co-authors of https://tools.ietf.org/html/draft-doron-dots-telemetry-00 draft and everyone who had contributed to that document.

Authors would like to thank Kaname Nishizuka, Jon Shallow, Wei Pan and Yuuhei Hayashi for comments and review.

15. References

15.1. Normative References

[Enterprise-Numbers]
"Private Enterprise Numbers", 2005, <http://www.iana.org/assignments/enterprise-numbers.html>.

[I-D.ietf-dots-data-channel]
Boucadair, M. and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-31 (work in progress), July 2019.

[I-D.ietf-dots-signal-call-home]
Reddy.K, T., Boucadair, M., and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Call Home", draft-ietf-dots-signal-call-home-08 (work in progress), March 2020.

[I-D.ietf-dots-signal-channel]
Reddy.K, T., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-41 (work in progress), January 2020.

[I-D.ietf-dots-signal-filter-control]
Nishizuka, K., Boucadair, M., Reddy.K, T., and T. Nagata, "Controlling Filtering Rules Using Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", draft-ietf-dots-signal-filter-control-03 (work in progress), March 2020.

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

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

15.2. Informative References

[I-D.ietf-dots-multihoming]
Boucadair, M., Reddy.K, T., and W. Pan, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-03 (work in progress), January 2020.

[I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-20 (work in progress), September 2019.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open Threat Signaling (DOTS) Requirements", RFC 8612, DOI 10.17487/RFC8612, May 2019, <https://www.rfc-editor.org/info/rfc8612>.

Authors' Addresses

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy (editor)
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: kondtir@gmail.com

Ehud Doron
Radware Ltd.
Raoul Wallenberg Street
Tel-Aviv 69710
Israel

Email: ehudd@radware.com

Meiling Chen
CMCC
32, Xuanwumen West
BeiJing, BeiJing 100053
China

Email: chenmeiling@chinamobile.com